Title
Measuring Generalization and Overfitting in Machine Learning

Author
Roelofs, Rebecca

Publication Date
2019

Peer reviewed|Thesis/dissertation

Measuring Generalization and Overfitting in Machine Learning

by

Rebecca Roelofs

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Benjamin Recht, Co-chair
Professor James Demmel, Co-chair
Associate Professor Moritz Hardt
Professor Bruno Olshausen

Summer 2019

Measuring Generalization and Overfitting in Machine Learning

Abstract

Measuring Generalization and Overfitting in Machine Learning

by

Rebecca Roelofs

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Benjamin Recht, Co-chair

Professor James Demmel, Co-chair

Due to the prevalence of machine learning algorithms and the potential for their decisions to profoundly impact billions of human lives, it is crucial that they are robust, reliable, and understandable. This thesis examines key theoretical pillars of machine learning surrounding generalization and overfitting, and tests the extent to which empirical behavior matches existing theory. We develop novel methods for measuring overfitting and generalization, and we characterize how reproducible observed behavior is across differences in optimization algorithm, dataset, task, evaluation metric, and domain.

First, we examine how optimization algorithms bias machine learning models towards solutions with varying generalization properties. We show that adaptive gradient methods empirically find solutions with inferior generalization behavior compared to those found by stochastic gradient descent. We then construct an example using a simple overparameterized model that corroborates the algorithms' empirical behavior on neural networks.

Next, we study the extent to which machine learning models have overfit to commonly reused datasets in both academic benchmarks and machine learning competitions. We build new test sets for the CIFAR-10 and ImageNet datasets and evaluate a broad range of classification models on the new datasets. All models experience a drop in accuracy, which indicates that current accuracy numbers are susceptible to even minute natural variations in the data distribution. Surprisingly, despite several years of adaptively selecting the models to perform well on these competitive benchmarks, we find no evidence of overfitting. We then analyze data from the machine learning platform Kaggle and find little evidence of substantial overfitting in ML competitions. These findings speak to the robustness of the holdout method across different data domains, loss functions, model classes, and human analysts.

Overall, our work suggests that the true concern for robust machine learning is distribution shift rather than overfitting, and designing models that still work reliably in dynamic environments is a challenging but necessary undertaking.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

First, thank you to my advisors Benjamin Recht and James Demmel. I am lucky to have studied under such intelligent, hard working, and creative scientists. Ben and I arrived at Berkeley at the same time, and his first topics class, *The Mathematics of Information and Data*, exposed me to interesting and challenging problems at the intersection of statistical machine learning and optimization, and eventually inspired me to change the course of my Ph.D. work. I also credit my interest and knowledge in parallel computing and numerical linear algebra to Jim's research agenda and courses.

Thank you to Shivaram Venkatarman for mentoring me during my early years of graduate school. Shivaram taught me the fundamentals of empirical research in machine learning, and his knowledge of systems and programming was indispensable. One of the most memorable bugs that we found together involved discovering that DORMQR was not thread-safe. Shivaram is now both a role model and a friend, and I truly admire his kindness, patience, and willingness to answer any question.

Thank you to my collaborators Sara Fridovich-Keil, Moritz Hardt, John Miller, Mitchell Stern, Ludwig Schmidt, Vaishaal Shankar, and Nathan Srebro, Stephen Tu, and Ashia Wilson. Without all of their hard work, this thesis would not be possible.

My experience at Berkeley was shaped by the people I was surrounded by. Overall, I found Berkeley to be a fun, collaborative environment, and it was a joy to go to work with some of the most brilliant and kind people I knew. Special thanks to Orianna DeMassi, Esther Rolf, Evan Sparks, Eric Jonas, Nikolai Matni, Ross Boczar, Stephen Tu, Horia Mania, Max Simchowitz, Sarah Dean, Lydia Liu, Karl Krauth, Wenshuo Guo, Vickie Ye, Ben Brock, Marquita Ellis, Michael Driscoll, Penporn Koanatakool, and William Kahan.

Lastly, thank you to my family who inspired me to pursue a Ph.D. and showed me that science could be an enjoyable and fruitful career. Yoyo especially deserves my utmost gratitude for listening to me rehearse talks over and over again, helping me navigate difficult professional situations, and editing drafts of this thesis. Above all, I am appreciative of the love and support I received from my family. This thesis is dedicated to them.

Chapter 1

Introduction

Over the past decade, an increasingly broad and diverse set of industries have deployed machine learning as a key component of their services. Technological innovations that transformed streams of data into fire hydrants, as well as ubiquitous economic pressures for automation, fueled the growing adoption of machine learning. Today, law enforcement, employment decisions, admissions, credit scoring, social networks, search results, and advertising all commonly use machine learning algorithms. Once deployed, these algorithms quickly achieve massive reach, and their decisions can potentially affect the lives of billions of people. In some application areas, such as medical diagnoses and self-driving cars, decisions made by machine learning algorithms can also have serious repercussions for human safety.

Since machine learning algorithms now have the power to shape and influence all aspects of society at unprecedented scale, it is critical that the algorithms are robust, reliable, and understandable. However, as we push the technology into more challenging application areas, weaknesses have emerged.

One shortcoming is that current classifiers are extremely sensitive to small shifts in the underlying data distribution. This fragility to distribution shift hinders the ability of the algorithm to generalize, or handle unseen or novel situations. For example, a self-driving car trained to drive on city streets would have difficulty driving on highways. Ultimately, the algorithm's lack of generalization causes it to make incorrect decisions, some of which have severe consequences. Moreover, distributional sensitivity leaves the underlying algorithms vulnerable to attack from malicious adversaries; by changing the underlying data in a way that is imperceptible to humans, an adversary can easily manipulate specific predictions made by the algorithm.

A related weakness of machine learning algorithms is that they are notoriously difficult to interpret. When mistakes inevitably occur, it is challenging to identify what aspect of the data or system caused the mistake and how one should fix the problem. Even among machine learning experts, there is a lack of understanding for how or why a machine learning system arrives at a certain decision, in part because much of the success of machine learning has been driven by empirical progress, with little guidance from theory. The majority of published papers have embraced a paradigm where the main justification for a new learning technique

is its improved performance on a few key benchmarks, yet there are few explanations as to *why* a proposed technique achieves a reliable improvement over prior work.

Deep neural networks, in particular, have proved difficult to analyze from a theoretical perspective. Several architectural components and optimization techniques—for example, batch normalization, residual connections, extreme overfitting, and increasing step sizes—work exceedingly well in practice but have little theoretical justification. While there has been some progress in analyzing these phenomena [82, 17, 18], our current theoretical understanding is not rich enough to predict practical behavior. As a result, our ability to propose novel innovations that improve existing networks is limited.

The goal of this thesis is to empirically examine key theoretical pillars of machine learning so that we can build algorithms that are more reliable and robust. If we can understand and identify precisely where the breakdowns between theory and practice occur, we can create a body of knowledge that is *reproducible* across many settings, giving us the tools we need to both recognize the limitations of existing algorithms and improve their ability to adapt to novel situations.

At a high level, the analysis we perform exhibits a common pattern: In each case, we isolate a key phenomenon, either originating from existing theory or "conventional wisdom", and then rigorously test the range of settings under which the phenomenon holds. We focus on how the phenomenon changes as we individually vary core parts of the machine learning system, such as the optimization algorithm, the data, the hypothesis class, and the task objective. We then verify how well the behavior we observe empirically matches existing theory.

One theme that arises in the thesis is that measurement matters. We cannot build our theoretical understanding of the principles that govern robustness and reliability without accurate measurements of generalization. One issue that arises immediately from the theoretical definition of generalization error is that the exact quantity of interest is impossible to evaluate because it requires knowing the underlying population distribution. We can use approximations to bypass this difficulty, but it is important to know what assumptions we use when we make these approximations and to be aware of situations where we break these assumptions. As we rely more and more on machine learning for real-world, safety-critical applications, our models must be robust to small shifts in the underlying data distribution. To achieve robustness, we must be able to measure it.

1.1 Formal background on generalization and overfitting

"Generalization" and "overfitting" are widely used throughout machine learning as umbrella terms; generalization is often interpreted as the broad ability of a classifier to handle new scenarios while overfitting is used to describe any unwanted performance drop of a machine learning model. In this section, we provide formal background that allows us to define more precisely the notions of generalization and overfitting that we use throughout the rest of the thesis.

1.1.1 Generalization error

In statistical learning theory, our goal is to predict an outcome y from a set $\mathcal{Y}$ of possible outcomes, given that we observe x from some feature space $\mathcal{X}$. Our input is a dataset of n labeled examples

$$\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \tag{1.1.1}$$

which we use to choose a function $f : \mathcal{X} \to \mathcal{Y}$ so that given a new pair (x, y), $f(x)$ is a good approximation to y. The classic approach to measuring how well $f(x)$ predicts y is to define a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which intuitively represents the cost of predicting $\hat{y} = f(x)$ when the true label is y.

We adopt the standard *probabilistic assumption* and posit the existence of a "true" underlying data distribution $\mathcal{D}$ over labeled examples (x, y). We assume that the pairs $(x_1, y_1), \ldots, (x_n, y_n)$ in our sample are chosen independently and identically from the data distribution $\mathcal{D}$. Then, we wish to choose f so that we minimize the expected population risk

$$L_{\mathcal{D}}(f) = \mathbb{E}[l(f(x), y)] \tag{1.1.2}$$

where the expectation is taken with respect to the draw of (x, y) from $\mathcal{D}$.

However, since we often do not know the exact form of the data distribution $\mathcal{D}$, we cannot always evaluate the expectation needed to compute the population risk. Instead, we use our sample $\mathcal{S}$ to evaluate the empirical loss

$$L_{\mathcal{S}}(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i). \tag{1.1.3}$$

We then minimize the empirical loss to learn a function $\hat{f}_n$ such that

$$\hat{f}_n = \arg\min_f L_{\mathcal{S}}(f) \tag{1.1.4}$$

We use the subscript n on $\hat{f}_n$ to denote explicitly that f depends on the sample $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Here, $\mathcal{S}$ is the training set, the empirical loss is akin to the training loss, and the process of finding the function $\hat{f}_n$ that minimizes the empirical loss is the process of training a model.

The *generalization error* G of $\hat{f}_n$ is the difference between the empirical loss and the population loss

$$G = L_{\mathcal{D}}(\hat{f}_n) - L_{\mathcal{S}}(\hat{f}_n). \tag{1.1.5}$$

Since we do not know how to evaluate $L_{\mathcal{D}}$, we rely on another sample $S_{\text{test}} \sim \mathcal{D}$ to estimate the population loss. Then, we compute an approximate generalization error $\hat{G}$ as

$$\hat{G} = L_{\mathcal{S}_{\text{test}}}(\hat{f}_n) - L_{\mathcal{S}}(\hat{f}_n). \tag{1.1.6}$$

In deep learning, a trained model often achieves a training loss of 0 (i.e. $L_{\mathcal{S}}(\hat{f}_n) = 0$), so we also sometimes assume that

$$\hat{G} \approx L_{\mathcal{S}_{\text{test}}}(\hat{f}_n) \tag{1.1.7}$$

In Chapter 2 we use the trained model's performance on the test set as a proxy for the model's generalization error. However, as we discuss next and revisit in Chapters 3 and 4, this approximation critically assumes that we have not used the test set to select the trained model.

1.1.2 Adaptive overfitting

In this thesis, we focus on *adaptive* overfitting, which is overfitting caused by test set reuse. When defining generalization error, we assumed that the model learned from the training set $\hat{f}_n$ does not depend on the test set $\mathcal{S}_{\text{test}}$. This assumption underlies essentially all empirical evaluations in machine learning since it allows us to argue that the model $\hat{f}_n$ generalizes. As long as the model $\hat{f}_n$ does not depend on the test set $\mathcal{S}_{\text{test}}$, standard concentration results [83] show that $L_{\mathcal{S}_{\text{test}}}(\hat{f}_n)$ is a good approximation of the true performance given by the population loss $L_{\mathcal{D}}(\hat{f}_n)$.

However, machine learning practitioners often undermine this assumption by selecting models and tuning hyperparameters based on the test loss. Especially when algorithm designers evaluate a large number of different models on the same test set, the final classifier may only perform well on the specific examples in the test set. The failure to generalize to the entire data distribution $\mathcal{D}$ manifests itself in a large *adaptivity gap* $L_{\mathcal{D}}(\hat{f}_n) - L_{\mathcal{S}_{\text{test}}}(\hat{f}_n)$ and leads to overly optimistic performance estimates.

In practice, if $\mathcal{S}_{\text{test}}$ has been used to select $\hat{f}_n$, we must draw a new test set $\mathcal{S}'_{\text{test}} \sim \mathcal{D}$ that is independent of $\hat{f}_n$ to evaluate the empirical loss as an approximation to the population loss $L_{\mathcal{D}}(\hat{f}_n)$. Then, we can empirically measure the amount of adaptive overfitting as the difference between the empirical loss on the new test set and the empirical loss on the original test set

$$L_{\mathcal{S}'_{\text{test}}}(\hat{f}_n) - L_{\mathcal{S}_{\text{test}}}(\hat{f}_n) \tag{1.1.8}$$

In Chapters 3 and 4 we exploit this strategy to explore to what extent adaptive overfitting occurs in popular machine learning benchmarks and competitions.

1.2 Dissertation overview

The goal of the thesis is to ensure that machine learning algorithms are robust and reliable. Our approach is to empirically evaluate key theoretical pillars of machine learning in order to better understand robustness. We explore how changing key components of a machine learning system, such as the dataset, the architecture, the task type, the evaluation metric, or the optimization algorithm, impact empirical measurements of generalization and overfitting.

First, Chapter 2 explores the impact of the optimization algorithm on generalization error. We demonstrate that adaptive gradient methods can find solutions that have worse generalization error when compared to the more traditional stochastic gradient descent.

Next, Chapter 3 explores the impact of the dataset on generalization error and adaptive overfitting. We create new test sets for CIFAR-10 and ImageNet that allow us to measure the

amount of adaptive overfitting on these popular benchmarks. Surprisingly, we find little to no evidence of adaptive overfitting despite the fact that these benchmarks have been reused intensively for almost a decade for model selection.

Finally, Chapter 4 explores the impact of both the task and evaluation metric when measuring adaptive overfitting in machine learning competitions. We conduct the first large meta-analysis of overfitting due to test set reuse in the machine learning community, analyzing over one hundred machine learning competitions on the Kaggle platform. Our longitudinal study shows, somewhat surprisingly, little evidence of substantial overfitting

Chapter 2

Generalization properties of adaptive gradient methods

Adaptive optimization methods, which perform local optimization with a metric constructed from the history of iterates, are becoming increasingly popular for training deep neural networks. Examples include AdaGrad, RMSProp, and Adam. In this chapter, we discuss the generalization properties of adaptive gradient methods and compare these to the more traditional stochastic gradient descent (SGD).

2.1 Introduction

An increasing share of deep learning researchers are training their models with *adaptive gradient methods* [19, 64] due to their rapid training time [46]. Adam [50] in particular has become the default algorithm used across many deep learning frameworks. However, the generalization and out-of-sample behavior of such adaptive gradient methods remains poorly understood. Given that many passes over the data are needed to minimize the training objective, typical regret guarantees do not necessarily ensure that the found solutions will generalize [77].

Notably, when the number of parameters exceeds the number of data points, it is possible that the choice of algorithm can dramatically influence which model is learned [69]. Given two different minimizers of some optimization problem, what can we say about their relative ability to generalize? In this paper, we show that adaptive and non-adaptive optimization methods indeed find very different solutions with very different generalization properties. We provide a simple generative model for binary classification where the population is linearly separable (i.e., there exists a solution with large margin), but AdaGrad [19], RMSProp [91], and Adam converge to a solution that incorrectly classifies new data with probability arbitrarily close to half. On this same example, SGD finds a solution with zero error on new data. Our construction shows that adaptive methods tend to give undue influence to spurious features that have no effect on out-of-sample generalization (defined as in 1.1.7).

We additionally present numerical experiments demonstrating that adaptive methods generalize less well than their non-adaptive counterparts. Our experiments reveal three primary findings. First, with the same amount of hyperparameter tuning, SGD and SGD with momentum outperform adaptive methods on the test set across all evaluated models and tasks. This is true even when the adaptive methods achieve the *same training loss or lower* than non-adaptive methods. Second, adaptive methods often display faster initial progress on the training set, but their performance quickly plateaus on the test set. Third, the same amount of tuning was required for all methods, including adaptive methods. This challenges the conventional wisdom that adaptive methods require less tuning. Moreover, as a useful guide to future practice, we propose a simple scheme for tuning learning rates and decays that performs well on all deep learning tasks we studied.

2.2 Background

The canonical optimization algorithms used to minimize risk are either stochastic gradient methods or stochastic momentum methods. Stochastic gradient methods can generally be written

$$w_{k+1} = w_k - \alpha_k \tilde{\nabla} f(w_k), \tag{2.2.1}$$

where $\tilde{\nabla} f(w_k) := \nabla f(w_k; x_{i_k})$ is the gradient of some loss function f computed on a batch of data x_{i_k}.

Stochastic momentum methods are a second family of techniques that have been used to accelerate training. These methods can generally be written as

$$w_{k+1} = w_k - \alpha_k \tilde{\nabla} f(w_k + \gamma_k(w_k - w_{k-1})) + \beta_k(w_k - w_{k-1}). \tag{2.2.2}$$

The sequence of iterates (2.2.2) includes Polyak's heavy-ball method (HB) with $\gamma_k = 0$, and Nesterov's Accelerated Gradient method (NAG) [85] with $\gamma_k = \beta_k$.

Notable exceptions to the general formulations (2.2.1) and (2.2.2) are adaptive gradient and adaptive momentum methods, which choose a local distance measure constructed using the entire sequence of iterates $(w_1, \cdots, w_k)$. These methods (including AdaGrad [19], RMSProp [91], and Adam [50]) can generally be written as

$$w_{k+1} = w_k - \alpha_k \mathrm{H}_k^{-1} \tilde{\nabla} f(w_k + \gamma_k(w_k - w_{k-1})) + \beta_k \mathrm{H}_k^{-1} \mathrm{H}_{k-1}(w_k - w_{k-1}), \tag{2.2.3}$$

where $\mathrm{H}_k := H(w_1, \cdots, w_k)$ is a positive definite matrix. Though not necessary, the matrix H_k is usually defined as

$$\mathrm{H}_k = \mathrm{diag}\left(\left\{ \sum_{i=1}^{k} \eta_i g_i \circ g_i \right\}^{1/2} \right), \tag{2.2.4}$$

where "$\circ$" denotes the entry-wise or Hadamard product, $g_k = \tilde{\nabla} f(w_k + \gamma_k(w_k - w_{k-1}))$, and η_k is some set of coefficients specified for each algorithm. That is, H_k is a diagonal matrix whose

entries are the square roots of a linear combination of squares of past gradient components. We will use the fact that H_k are defined in this fashion in the sequel. For the specific settings of the parameters for many of the algorithms used in deep learning, see Table 2.1. Adaptive methods attempt to adjust an algorithm to the geometry of the data. In contrast, stochastic gradient descent and related variants use the ℓ_2 geometry inherent to the parameter space, and are equivalent to setting $\mathrm{H}_k = \mathrm{I}$ in the adaptive methods.

	SGD	HB	NAG	AdaGrad	RMSProp	Adam
G_k	I	I	I	$\mathrm{G_{k-1}} + \mathrm{D}_k$	$\beta_2 \mathrm{G_{k-1}} + (1-\beta_2)\mathrm{D}_k$	$\frac{\beta_2}{1-\beta_2^k}\mathrm{G_{k-1}} + \frac{(1-\beta_2)}{1-\beta_2^k}\mathrm{D}_k$
α_k	α	α	α	α	α	$\alpha\frac{1-\beta_1}{1-\beta_1^k}$
β_k	0	β	β	0	0	$\frac{\beta_1(1-\beta_1^{k-1})}{1-\beta_1^k}$
γ	0	0	β	0	0	0

Table 2.1: Parameter settings of optimization algorithms used in deep learning. Here, $\mathrm{D}_k = \mathrm{diag}(g_k \circ g_k)$ and $\mathrm{G}_k := \mathrm{H}_k \circ \mathrm{H}_k$. We omit the additional ϵ added to the adaptive methods, which is only needed to ensure non-singularity of the matrices H_k.

In this context, *generalization* refers to the performance of a solution w on a broader population. Performance is often defined in terms of a different loss function than the function f used in training. For example, in classification tasks, we typically define generalization in terms of classification error rather than cross-entropy.

2.2.1 Related work

Understanding how optimization relates to generalization is a very active area of current machine learning research. Most of the seminal work in this area has focused on understanding how early stopping can act as implicit regularization [100]. In a similar vein, Ma and Belkin [60] have shown that gradient methods may not be able to find complex solutions at all in any reasonable amount of time. Hardt et al. [77] show that SGD is uniformly stable, and therefore solutions with low training error found quickly will generalize well. Similarly, using a stability argument, Raginsky et al. [74] have shown that Langevin dynamics can find solutions that generalize better than ordinary SGD in non-convex settings. Neyshabur, Srebro, and Tomioka [69] discuss how algorithmic choices can act as implicit regularizer. In a similar vein, Neyshabur, Salakhutdinov, and Srebro [68] show that a different algorithm, one which performs descent using a metric that is invariant to re-scaling of the parameters, can lead to solutions which sometimes generalize better than SGD. Our work supports the work of [68] by drawing connections between the metric used to perform local optimization and the ability of the training algorithm to find solutions that generalize. However, we focus primarily on the different generalization properties of adaptive and non-adaptive methods.

A similar line of inquiry has been pursued by Keskar et al. [49]. Horchreiter and Schmidhuber [36] showed that "sharp" minimizers generalize poorly, whereas "flat" minimizers gen-

eralize well. Keskar et al. empirically show that Adam converges to sharper minimizers when the batch size is increased. However, they observe that even with small batches, Adam does not find solutions whose performance matches state-of-the-art. In the current work, we aim to show that the choice of Adam as an optimizer itself strongly influences the set of minimizers that any batch size will ever see, and help explain why they were unable to find solutions that generalized particularly well.

2.3 The perils of preconditioning

The goal of this section is to illustrate the following observation: *when a problem has multiple global minima, different algorithms can find entirely different solutions*. In particular, we will show that adaptive gradient methods might find very poor solutions. To simplify the presentation, let us restrict our attention to the simple binary least-squares classification problem, where we can easily compute closed form formulae for the solutions found by different methods. In least-squares classification, we aim to solve

$$\text{minimize}_w \quad R_S[w] := \|Xw - y\|_2^2. \tag{2.3.1}$$

Here X is an $n \times d$ matrix of features and y is an n-dimensional vector of labels in $\{-1, 1\}$. We aim to find the best linear classifier w. Note that when $d > n$, if there is a minimizer with loss 0 then there is an infinite number of global minimizers. The question remains: what solution does an algorithm find and how well does it generalize to unseen data?

2.3.1 Non-adaptive methods

Most common methods when applied to (2.3.1) will find the same solution. Indeed, any gradient or stochastic gradient of R_S must lie in the span of the rows of X. Therefore, any method that is initialized in the row span of X (say, for instance at $w = 0$) and uses only linear combinations of gradients, stochastic gradients, and previous iterates must also lie in the row span of X. The unique solution that lies in the row span of X also happens to be the solution with minimum Euclidean norm. We thus denote $w^{\text{SGD}} = X^T(XX^T)^{-1}y$. Almost all non-adaptive methods like SGD, SGD with momentum, mini-batch SGD, gradient descent, Nesterov's method, and the conjugate gradient method will converge to this minimum norm solution. Minimum norm solutions have the largest *margin*, or distance between the decision boundary and the closest data point to the decisioun boundary, out of all solutions of the equation $Xw = y$. Maximizing margin has a long and fruitful history in machine learning, and thus it is a pleasant surprise that gradient descent naturally finds a max-margin solution.

2.3.2 Adaptive methods

Let us now consider the case of adaptive methods, restricting our attention to diagonal adaptation. While it is difficult to derive the general form of the solution, we can analyze special

cases. Indeed, we can construct a variety of instances where adaptive methods converge to solutions with low ℓ_∞ norm rather than low ℓ_2 norm.

For a vector $x \in \mathbb{R}^q$, let $\text{sign}(x)$ denote the function that maps each component of x to its sign.

Lemma 2.3.1. *Suppose $X^T y$ has no components equal to 0 and there exists a scalar c such that $X\ \text{sign}(X^T y) = cy$. Then, when initialized at $w_0 = 0$, AdaGrad, Adam, and RMSProp all converge to the unique solution $w \propto \text{sign}(X^T y)$.*

In other words, whenever there exists a solution of $Xw = y$ that is proportional to $\text{sign}(X^T y)$, this is precisely the solution to where all of the adaptive gradient methods converge.

Proof. We prove this lemma by showing that the entire trajectory of the algorithm consists of iterates whose components have constant magnitude. In particular, we will show that

$$w_k = \lambda_k\ \text{sign}(X^T y)\,.$$

for some scalar λ_k. Note that $w_0 = 0$ satisfies the assertion with $\lambda_0 = 0$.

Now, assume the assertion holds for all $k \le t$. Observe that

$$\begin{aligned}
\nabla R_S(w_k + \gamma_k(w_k - w_{k-1})) &= X^T(X(w_k + \gamma_k(w_k - w_{k-1})) - y) \\
&= X^T\left\{(\lambda_k + \gamma_k(\lambda_k - \lambda_{k-1}))X\ \text{sign}(X^T y) - y\right\} \\
&= \left\{(\lambda_k + \gamma_k(\lambda_k - \lambda_{k-1}))c - 1\right\} X^T y \\
&= \mu_k X^T y,
\end{aligned}$$

where the last equation defines μ_k. Hence, letting $g_k = \nabla R_S(w_k + \gamma_k(w_k - w_{k-1}))$, we also have

$$\mathrm{H}_k = \text{diag}\left(\left\{\sum_{s=1}^{k} \eta_s\ g_s \circ g_s\right\}^{1/2}\right) = \text{diag}\left(\left\{\sum_{s=1}^{k} \eta_s \mu_s^2\right\}^{1/2} |X^T y|\right) = \nu_k\,\text{diag}\left(|X^T y|\right)$$

where $|u|$ denotes the component-wise absolute value of a vector and the last equation defines ν_k.

Thus we have,

$$\begin{aligned}
w_{k+1} &= w_k - \alpha_k \mathrm{H}_k^{-1} \nabla f(w_k + \gamma_k(w_k - w_{k-1})) + \beta_t \mathrm{H}_k^{-1} \mathrm{H}_{k-1}(w_k - w_{k-1}) && (2.3.2) \\
&= \left\{\lambda_k - \frac{\alpha_k \mu_k}{\nu_k} + \frac{\beta_k \nu_{k-1}}{\nu_k}(\lambda_k - \lambda_{k-1})\right\} \text{sign}(X^T y) && (2.3.3)
\end{aligned}$$

proving the claim. □

Note that this solution w could be obtained without any optimization at all. One simply could subtract the means of the positive and negative classes and take the sign of the resulting vector. This solution is far simpler than the one obtained by gradient methods, and it would be surprising if such a simple solution would perform particularly well. We now turn to showing that such solutions can indeed generalize arbitrarily poorly.

2.3.3 Adaptivity can overfit

Lemma 2.3.1 allows us to construct a particularly pernicious generative model where AdaGrad fails to find a solution that generalizes. This example uses infinite dimensions to simplify bookkeeping, but one could take the dimensionality to be $6n$. Note that in deep learning, we often have a number of parameters equal to $25n$ or more [90], so this is not a particularly high dimensional example by contemporary standards. For $i = 1, \ldots, n$, sample the label y_i to be 1 with probability p and -1 with probability $1 - p$ for some $p > 1/2$. Let x be an infinite dimensional vector with entries

$$x_{ij} = \begin{cases} y_i & j = 1 \\ 1 & j = 2, 3 \\ 1 & j = 4 + 5(i-1), \ldots, 4 + 5(i-1) + 2(1 - y_i) \\ 0 & \text{otherwise} \end{cases}.$$

In other words, the first feature of x_i is the class label. The next 2 features are always equal to 1. After this, there is a set of features *unique to* x_i that are equal to 1. If the class label is 1, then there is 1 such unique feature. If the class label is -1, then there are 5 such features. Note that for such a data set, the only discriminative feature is the first one! Indeed, one can perform perfect classification using only the first feature. The other features are all useless. Features 2 and 3 are constant, and each of the remaining features only appear for one example in the data set. However, as we will see, algorithms without such *a priori* knowledge may not be able to learn these distinctions.

Take n samples and consider the AdaGrad solution to the minimizing $||Xw - y||^2$. First we show that the conditions of Lemma 2.3.1 hold. Let $b = \sum_{i=1}^{n} y_i$ and assume for the sake of simplicity that $b > 0$. This will happen with arbitrarily high probability for large enough n. Define $u = X^T y$ and observe that

$$u_j = \begin{cases} n & j = 1 \\ b & j = 2, 3 \\ y_j & \text{if } j > 3 \text{ and } x_j = 1 \end{cases} \qquad \text{and} \qquad \operatorname{sign}(u_j) = \begin{cases} 1 & j = 1 \\ 1 & j = 2, 3 \\ y_j & \text{if } j > 3 \text{ and } x_j = 1 \end{cases}$$

Thus we have $\langle u, x_i \rangle = y_i + 2 + y_i(3 - 2y_i) = 4y_i$, as desired. Hence, the AdaGrad solution $w^{\text{ada}} \propto \operatorname{sign}(u)$. In particular, w^{ada} has all of its components either equal to 0 or to $\pm\tau$ for some positive constant τ. Now since w^{ada} has the same sign pattern as u, the first three components of w_{ada} are equal to each other. But for a new data point, x^{test}, the only features that are nonzero in both x^{test} and w^{ada} are the first three. In particular, we have

$$\langle w^{\text{ada}}, x^{\text{test}} \rangle = \tau(y^{(test)} + 2) > 0\,.$$

Therefore, the AdaGrad solution will label all unseen data as being in the positive class!

Now let's turn to the minimum norm solution. Let $\mathcal{P}$ and $\mathcal{N}$ denote the set of positive and negative examples respectively. Let $n_+ = |\mathcal{P}|$ and $n_- = |\mathcal{N}|$. By symmetry, we have

that the minimum norm solution will have the form $w^{\text{SGD}} = \sum_{i \in \mathcal{P}} \alpha_+ x_i - \sum_{j \in \mathcal{N}} \alpha_- x_j$ for some nonnegative scalars α_+ and α_-. These scalars can be found by solving $XX^T\alpha = y$. In closed form we have

$$\alpha_+ = \frac{4n_- + 3}{9n_+ + 3n_- + 8n_+n_- + 3} \qquad \text{and} \qquad \alpha_- = \frac{4n_+ + 1}{9n_+ + 3n_- + 8n_+n_- + 3}\,. \tag{2.3.4}$$

The algebra required to compute these coefficients can be found in Section 2.3.4. For a new data point, x^{test}, again the only features that are nonzero in both x^{test} and w^{SGD} are the first three. Thus we have

$$\langle w^{\text{SGD}}, x^{\text{test}} \rangle = y^{\text{test}}(n_+\alpha_+ + n_-\alpha_-) + 2(n_+\alpha_+ - n_-\alpha_-)\,.$$

Using (2.3.4), we see that whenever $n_+ > n_-/3$, the SGD solution makes no errors.

Though this generative model was chosen to illustrate extreme behavior, it shares salient features of many common machine learning instances. There are a few frequent features, where some predictor based on them is a good predictor, though these might not be easy to identify from first inspection. Additionally, there are many other features which are very sparse. On finite training data it looks like such features are good for prediction, since each such feature is very discriminatory for a particular training example, but this is over-fitting and an artifact of having fewer training examples then features. Moreover, we will see shortly that adaptive methods typically generalize worse than their non-adaptive counterparts on real datasets as well.

2.3.4 Why SGD converges to the minimum norm solution

The simplest derivation of the minimum norm solution uses the kernel trick. We know that the optimal solution has the form $w^{\text{SGD}} = X^T\alpha$ where $\alpha = K^{-1}y$ and $K = XX^T$. Note that

$$K_{ij} = \begin{cases} 4 & \text{if } i = j \text{ and } y_i = 1 \\ 8 & \text{if } i = j \text{ and } y_i = -1 \\ 3 & \text{if } i \neq j \text{ and } y_iy_j = 1 \\ 1 & \text{if } i \neq j \text{ and } y_iy_j = -1 \end{cases}$$

Positing that $\alpha_i = \alpha_+$ if $y_i = 1$ and $\alpha + i = \alpha_-$ if $y_i = -1$ leaves us with the equation

$$\begin{aligned} (3n_+ + 1)\alpha_+ - n_-\alpha_- &= 1 \\ -n_+\alpha_+ + (3n_- + 3)\alpha_- &= 1 \end{aligned}$$

Solving this system of equations yields (2.3.4).

Name	Network type	Architecture	Dataset	Framework
C1	Deep Convolutional	`cifar.torch`	CIFAR-10	Torch
L1	2-Layer LSTM	`torch-rnn`	War & Peace	Torch
L2	2-Layer LSTM + Feedforward	`span-parser`	Penn Treebank	DyNet
L3	3-Layer LSTM	`emnlp2016`	Penn Treebank	Tensorflow

Table 2.2: Summary of the model architectures, datasets, and frameworks used in deep learning experiments. [1]

2.4 Deep learning experiments

Having established that adaptive and non-adaptive methods can find quite different solutions in the convex setting, we now turn to an empirical study of deep neural networks to see whether we observe a similar discrepancy in generalization. We compare two non-adaptive methods – SGD and the heavy ball method (HB) – to three popular adaptive methods – AdaGrad, RMSProp and Adam. We study performance on four deep learning problems: **(C1)** the CIFAR-10 image classification task, **(L1)** character-level language modeling on the novel War and Peace, and **(L2)** discriminative parsing and **(L3)** generative parsing on Penn Treebank. In the interest of reproducibility, we use a network architecture for each problem that is either easily found online (C1, L1, L2, and L3) or produces state-of-the-art results (L2 and L3). Table 2.2 summarizes the setup for each application. We take care to make minimal changes to the architectures and their data pre-processing pipelines in order to best isolate the effect of each optimization algorithm.

We conduct each experiment 5 times from randomly initialized starting points, using the initialization scheme specified in each code repository. We allocate a pre-specified budget on the number of epochs used for training each model. When a development set was available, we chose the settings that achieved the best peak performance on the development set by the end of the fixed epoch budget. CIFAR-10 did not have an explicit development set, so we chose the settings that achieved the lowest training loss at the end of the fixed epoch budget.

Our experiments show the following primary findings: (*i*) Adaptive methods find solutions that generalize worse than those found by non-adaptive methods. (*ii*) Even when the adaptive methods achieve the *same training loss or lower* than non-adaptive methods, the development or test performance is worse. (*iii*) Adaptive methods often display faster initial progress on the training set, but their performance quickly plateaus on the development set. (*iv*) Though conventional wisdom suggests that Adam does not require tuning, we find that tuning the initial learning rate and decay scheme for Adam yields significant improvements over its default settings in all cases.

[1] Architectures can be found at the following links: (1) `cifar.torch`: `https://github.com/szagoruyko/cifar.torch`; (2) `torch-rnn`: `https://github.com/jcjohnson/torch-rnn`; (3) `span-parser`: `https://github.com/jhcross/span-parser`; (4) `emnlp2016`: `https://github.com/cdg720/emnlp2016`.

2.4.1 Hyperparameter tuning

Optimization hyperparameters have a large influence on the quality of solutions found by optimization algorithms for deep neural networks. The algorithms under consideration have many hyperparameters: the initial step size α_0, the step decay scheme, the momentum value β_0, the momentum schedule β_k, the smoothing term ϵ, the initialization scheme for the gradient accumulator, and the parameter controlling how to combine gradient outer products, to name a few. A grid search on a large space of hyperparameters is infeasible even with substantial industrial resources, and we found that the parameters that impacted performance the most were the initial step size and the step decay scheme. We left the remaining parameters with their default settings. We describe the differences between the default settings of Torch, DyNet, and Tensorflow in Section 2.6.1 for completeness.

To tune the step sizes, we evaluated a logarithmically-spaced grid of five step sizes. If the best performance was ever at one of the extremes of the grid, we would try new grid points so that the best performance was contained in the middle of the parameters. For example, if we initially tried step sizes 2, 1, 0.5, 0.25, and 0.125 and found that 2 was the best performing, we would have tried the step size 4 to see if performance was improved. If performance improved, we would have tried 8 and so on. We list the initial step sizes we tried in Section 2.6.2.

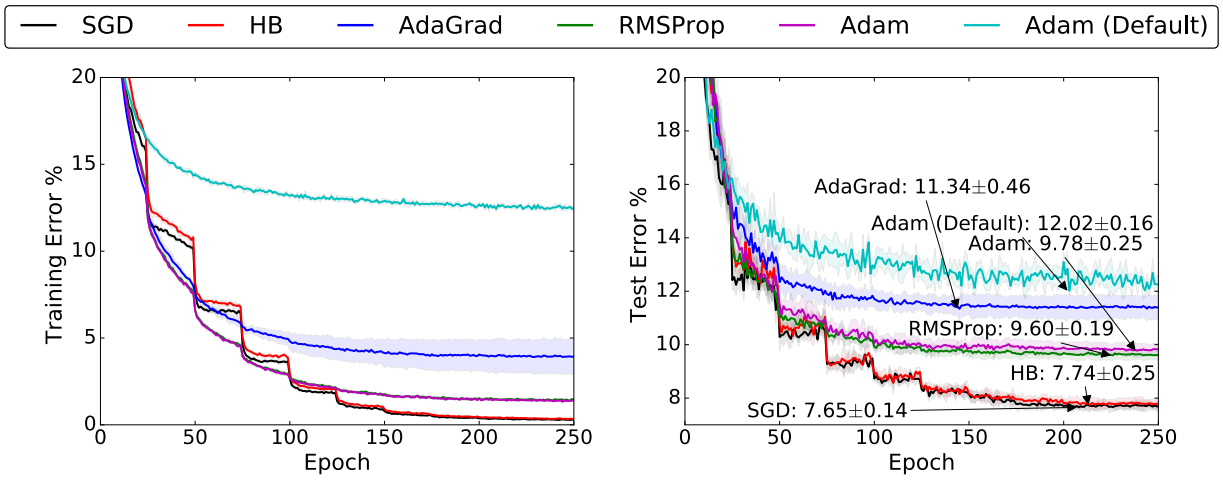
For step size decay, we explored two separate schemes, a development-based decay scheme (dev-decay) and a fixed frequency decay scheme (fixed-decay). For dev-decay, we keep track of the best validation performance so far, and at each epoch decay the learning rate by a constant factor δ if the model does not attain a new best value. For fixed-decay, we decay the learning rate by a constant factor δ every k epochs. We recommend the dev-decay scheme when a development set is available; not only does it have fewer hyperparameters than the fixed frequency scheme, but our experiments also show that it produces results comparable to, or better than, the fixed-decay scheme.

2.4.2 Convolutional neural network

We used the VGG+BN+Dropout network for CIFAR-10 from the Torch blog [101], which in prior work achieves a baseline test error of 7.55%. Figure 2.1 shows the learning curve for each algorithm on both the training and test dataset.

We observe that the solutions found by SGD and HB do indeed generalize better than those found by adaptive methods. The best overall test error found by a non-adaptive algorithm, SGD, was $7.65 \pm 0.14\%$, whereas the best adaptive method, RMSProp, achieved a test error of $9.60 \pm 0.19\%$.

Early on in training, the adaptive methods appear to be performing better than the non-adaptive methods, but starting at epoch 50, even though the training error of the adaptive methods is still lower, SGD and HB begin to outperform adaptive methods on the test error. By epoch 100, the performance of SGD and HB surpass all adaptive methods on both train and test. Among all adaptive methods, AdaGrad's rate of improvement flatlines the



(a) CIFAR-10 (Train) (b) CIFAR-10 (Test)

Figure 2.1: Training (left) and top-1 test error (right) on CIFAR-10. The annotations indicate where the best performance is attained for each method. The shading represents ± one standard deviation computed across five runs from random initial starting points. In all cases, adaptive methods are performing worse on both train and test than non-adaptive methods.

earliest. We also found that by increasing the step size, we could drive the performance of the adaptive methods down in the first 50 or so epochs, but the aggressive step size made the flatlining behavior worse, and no step decay scheme could fix the behavior.

2.4.3 Character-Level language modeling

Using the `torch-rnn` library, we train a character-level language model on the text of the novel War and Peace, running for a fixed budget of 200 epochs. Our results are shown in Figures 2.2a and 2.2b.

Under the fixed-decay scheme, the best configuration for all algorithms except AdaGrad was to decay relatively late with regards to the total number of epochs, either 60 or 80% through the total number of epochs and by a large amount, dividing the step size by 10. The dev-decay scheme paralleled (within the same standard deviation) the results of the exhaustive search over the decay frequency and amount; we report the curves from the fixed policy.

Overall, SGD achieved the lowest test loss at 1.212 ± 0.001. AdaGrad has fast initial progress, but flatlines. The adaptive methods appear more sensitive to the initialization scheme than non-adaptive methods, displaying a higher variance on both train and test. Surprisingly, RMSProp closely trails SGD on test loss, confirming that it is not impossible for adaptive methods to find solutions that generalize well. We note that there are step

configurations for RMSProp that drive the training loss below that of SGD, but these configurations cause erratic behavior on test, driving the test error of RMSProp above Adam.

2.4.4 Constituency parsing

A constituency parser is used to predict the hierarchical structure of a sentence, breaking it down into nested clause-level, phrase-level, and word-level units. We carry out experiments using two state-of-the-art parsers: the stand-alone discriminative parser of Cross and Huang [12], and the generative reranking parser of Choe and Charniak [7]. In both cases, we use the dev-decay scheme with $\delta = 0.9$ for learning rate decay.

Discriminative model. Cross and Huang [12] develop a transition-based framework that reduces constituency parsing to a sequence prediction problem, giving a one-to-one correspondence between parse trees and sequences of structural and labeling actions. Using their code with the default settings, we trained for 50 epochs on the Penn Treebank [63], comparing labeled F1 scores on the training and development data over time. RMSProp was not implemented in the used version of DyNet, and we omit it from our experiments. Results are shown in Figures 2.2c and 2.2d.

We find that SGD obtained the best overall performance on the development set, followed closely by HB and Adam, with AdaGrad trailing far behind. The default configuration of Adam without learning rate decay actually achieved the best overall training performance by the end of the run, but was notably worse than tuned Adam on the development set.

Interestingly, Adam achieved its best development F1 of 91.11 quite early, after just 6 epochs, whereas SGD took 18 epochs to reach this value and didn't reach its best F1 of 91.24 until epoch 31. On the other hand, Adam continued to improve on the training set well after its best development performance was obtained, while the peaks for SGD were more closely aligned.

Generative model. Choe and Charniak [7] show that constituency parsing can be cast as a language modeling problem, with trees being represented by their depth-first traversals. This formulation requires a separate base system to produce candidate parse trees, which are then rescored by the generative model. Using an adapted version of their code base,[2] we retrained their model for 100 epochs on the Penn Treebank. However, to reduce computational costs, we made two minor changes: (a) we used a smaller LSTM hidden dimension of 500 instead of 1500, finding that performance decreased only slightly; and (b) we accordingly lowered the dropout ratio from 0.7 to 0.5. Since they demonstrated a high correlation between perplexity (the exponential of the average loss) and labeled F1 on the

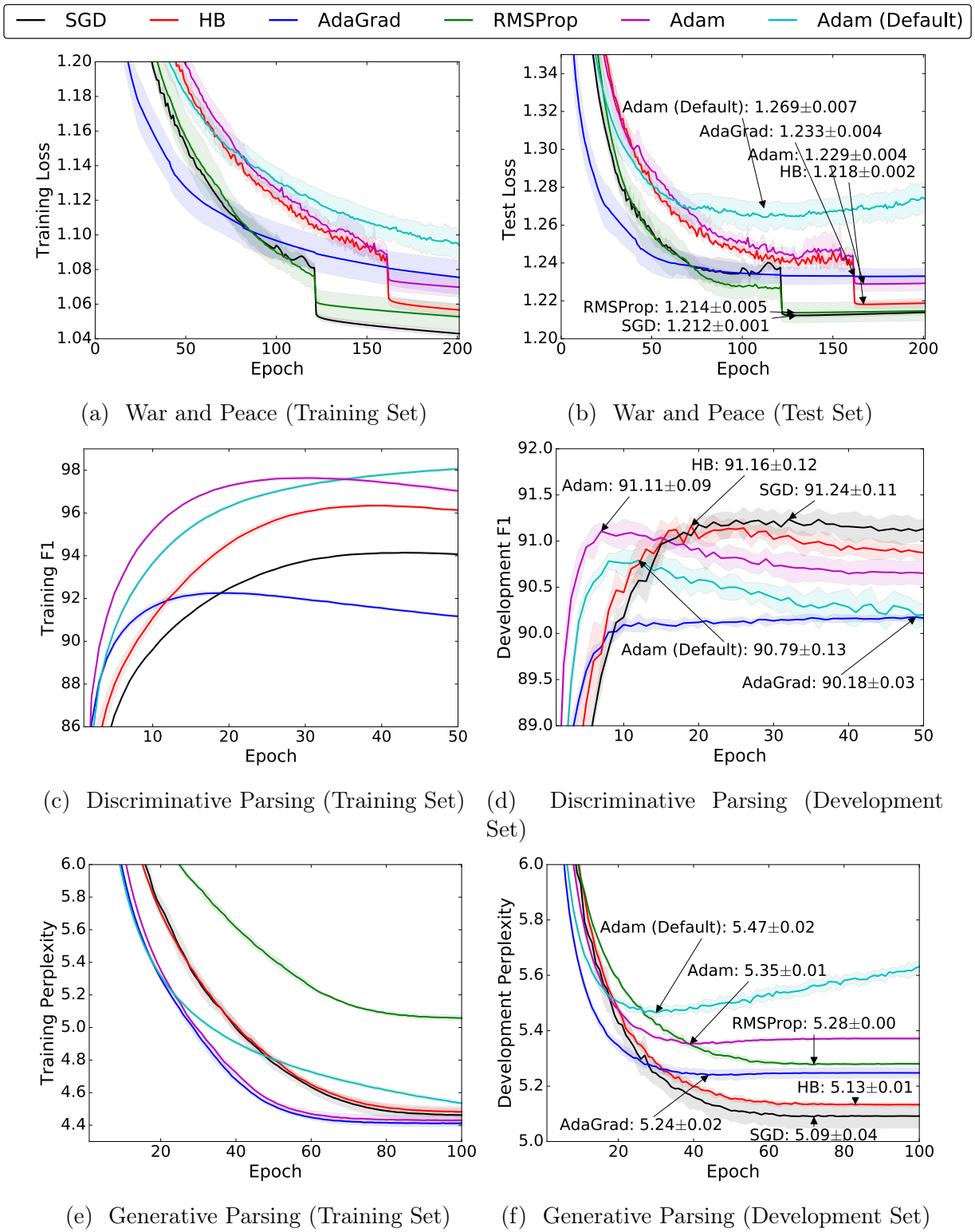
[2]While the code of Choe and Charniak treats the entire corpus as a single long example, relying on the network to reset itself upon encountering an end-of-sentence token, we use the more conventional approach of resetting the network for each example. This reduces training efficiency slightly when batches contain examples of different lengths, but removes a potential confounding factor from our experiments.

development set, we explored the relation between training and development perplexity to avoid any conflation with the performance of a base parser.

Our results are shown in Figures 2.2e and 2.2f. On development set performance, SGD and HB obtained the best perplexities, with SGD slightly ahead. Despite having one of the best performance curves on the training dataset, Adam achieves the worst development perplexities.

2.5 Conclusion

Despite the fact that our experimental evidence demonstrates that adaptive methods are not advantageous for machine learning, the Adam algorithm remains incredibly popular. We are not sure exactly as to why, but hope that our step-size tuning suggestions make it easier for practitioners to use standard stochastic gradient methods in their research. In our conversations with other researchers, we have surmised that adaptive gradient methods are particularly popular for training GANs [79, 43] and Q-learning with function approximation [66, 56]. Both of these applications stand out because they are not solving optimization problems. It is possible that the dynamics of Adam are accidentally well matched to these sorts of optimization-free iterative search procedures. It is also possible that carefully tuned stochastic gradient methods may work as well or better in both of these applications. It is an exciting direction of future work to determine which of these possibilities is true and to understand better as to why.



(a) War and Peace (Training Set)

(b) War and Peace (Test Set)

(c) Discriminative Parsing (Training Set)

(d) Discriminative Parsing (Development Set)

(e) Generative Parsing (Training Set)

(f) Generative Parsing (Development Set)

Figure 2.2: Performance curves on the training data (left) and the development/test data (right) for three experiments on natural language tasks. The annotations indicate where the best performance is attained for each method. The shading represents one standard deviation computed across five runs from random initial starting points.

2.6 Supplementary material

2.6.1 Differences between Torch, DyNet, and Tensorflow

	Torch	Tensorflow	DyNet
SGD Momentum	0	No default	0.9
AdaGrad Initial Mean	0	0.1	0
AdaGrad ϵ	1e-10	Not used	1e-20
RMSProp Initial Mean	0	1.0	–
RMSProp β	0.99	0.9	–
RMSProp ϵ	1e-8	1e-10	–
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999

Table 2.3: Default hyperparameters for algorithms in deep learning frameworks.

Table 2.3 lists the default values of the parameters for the various deep learning packages used in our experiments. In Torch, the Heavy Ball algorithm is callable simply by changing default momentum away from 0 with `nesterov=False`. In Tensorflow and DyNet, SGD with momentum is implemented separately from ordinary SGD. For our Heavy Ball experiments we use a constant momentum of $\beta = 0.9$.

2.6.2 Step sizes used for parameter tuning

CIFAR-10

- SGD: {2, 1, 0.5 (best), 0.25, 0.05, 0.01}
- HB: {2, 1, 0.5 (best), 0.25, 0.05, 0.01}
- AdaGrad: {0.1, 0.05, 0.01 (best, def.), 0.0075, 0.005}
- RMSProp: {0.005, 0.001, 0.0005, 0.0003 (best), 0.0001}
- Adam: {0.005, 0.001 (default), 0.0005, 0.0003 (best), 0.0001, 0.00005}

The default Torch step sizes for SGD (0.001) , HB (0.001), and RMSProp (0.01) were outside the range we tested.

War & Peace

- SGD: {2, 1 (best), 0.5, 0.25, 0.125}
- HB: {2, 1 (best), 0.5, 0.25, 0.125}

- AdaGrad: {0.4, 0.2, 0.1, 0.05 (best), 0.025}
- RMSProp: {0.02, 0.01, 0.005, 0.0025, 0.00125, 0.000625, 0.0005 (best), 0.0001}
- Adam: {0.005, 0.0025, 0.00125, 0.000625 (best), 0.0003125, 0.00015625}

Under the fixed-decay scheme, we selected learning rate decay frequencies from the set $\{10, 20, 40, 80, 120, 160, \infty\}$ and learning rate decay amounts from the set $\{0.1, 0.5, 0.8, 0.9\}$.

Discriminative Parsing

- SGD: {1.0, 0.5, 0.2, 0.1 (best), 0.05, 0.02, 0.01}
- HB: {1.0, 0.5, 0.2, 0.1, 0.05 (best), 0.02, 0.01, 0.005, 0.002}
- AdaGrad: {1.0, 0.5, 0.2, 0.1, 0.05, 0.02 (best), 0.01, 0.005, 0.002, 0.001, 0.0005, 0.0002, 0.0001}
- RMSProp: Not implemented in DyNet.
- Adam: {0.01, 0.005, 0.002 (best), 0.001 (default), 0.0005, 0.0002, 0.0001}

Generative Parsing

- SGD: {1.0, 0.5 (best), 0.25, 0.1, 0.05, 0.025, 0.01}
- HB: {0.25, 0.1, 0.05, 0.02, 0.01 (best), 0.005, 0.002, 0.001}
- AdaGrad: {5.0, 2.5, 1.0, 0.5, 0.25 (best), 0.1, 0.05, 0.02, 0.01}
- RMSProp: {0.05, 0.02, 0.01, 0.005, 0.002 (best), 0.001, 0.0005, 0.0002, 0.0001}
- Adam: {0.005, 0.001, 0.001 (default), 0.0005 (best), 0.0002, 0.0001}

Chapter 3

Do ImageNet classifiers generalize to ImageNet?

3.1 Introduction

The overarching goal of machine learning is to produce models that *generalize*. We usually quantify generalization by measuring the performance of a model on a held-out test set. What does good performance on the test set then imply? At the very least, one would hope that the model also performs well on a new test set assembled from the same data source by following the same data cleaning protocol.

In this chapter, we realize this thought experiment by replicating the dataset creation process for two prominent benchmarks, CIFAR-10 and ImageNet [53, 15]. In contrast to the ideal outcome, we find that a wide range of classification models fail to reach their original accuracy scores. The accuracy drops range from 3% to 15% on CIFAR-10 and 11% to 14% on ImageNet. On ImageNet, the accuracy loss amounts to approximately five years of progress in a highly active period of machine learning research.

Conventional wisdom suggests that such drops arise because the models have been adapted to the specific images in the original test sets, e.g., via extensive hyperparameter tuning. However, our experiments show that the relative order of models is almost exactly preserved on our new test sets: the models with highest accuracy on the original test sets are still the models with highest accuracy on the new test sets. Moreover, there are no diminishing returns in accuracy. In fact, every percentage point of accuracy improvement on the original test set translates to a *larger* improvement on our new test sets. So although later models could have been adapted more to the test set, they see smaller drops in accuracy. These results provide evidence that exhaustive test set evaluations are an effective way to improve image classification models. Adaptivity is therefore an unlikely explanation for the accuracy drops.

Instead, we propose an alternative explanation based on the relative difficulty of the original and new test sets. We demonstrate that it is possible to recover the original ImageNet

accuracies almost exactly if we only include the easiest images from our candidate pool. This suggests that the accuracy scores of even the best image classifiers are still highly sensitive to minutiae of the data cleaning process. This brittleness puts claims about human-level performance into context [47, 33, 81]. It also shows that current classifiers still do not generalize reliably even in the benign environment of a carefully controlled reproducibility experiment.

Figure 3.1 shows the main result of our experiment. Before we describe our methodology in Section 3.3, the next section provides relevant background. To enable future research, we release both our new test sets and the corresponding code.[1]

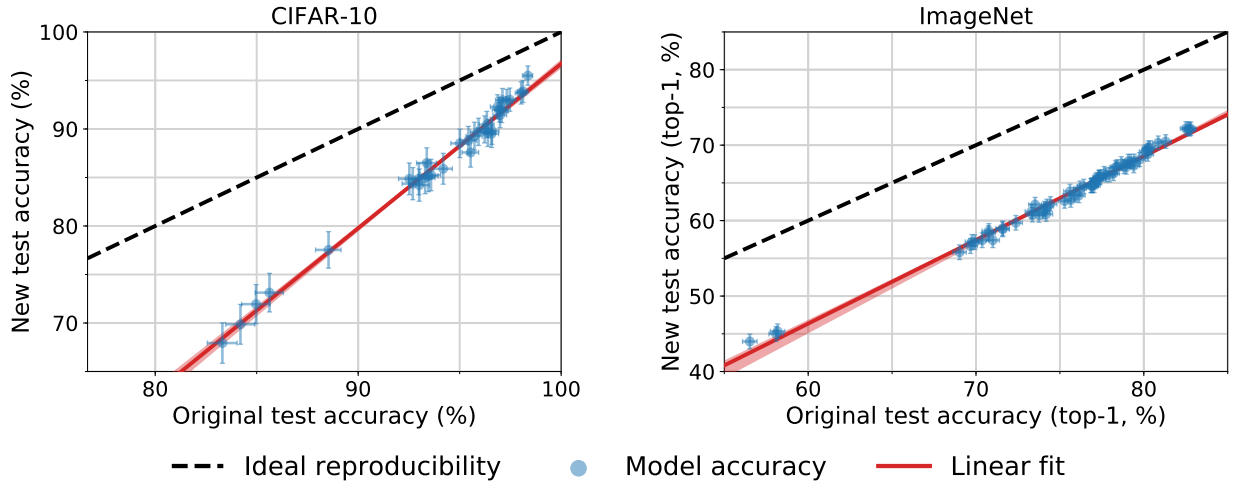


Figure 3.1: Model accuracy on the original test sets vs. our new test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The plots reveal two main phenomena: (i) There is a significant drop in accuracy from the original to the new test sets. (ii) The model accuracies closely follow a linear function with slope *greater* than 1 (1.7 for CIFAR-10 and 1.1 for ImageNet). This means that every percentage point of progress on the original test set translates into more than one percentage point on the new test set. The two plots are drawn so that their aspect ratio is the same, i.e., the slopes of the lines are visually comparable. The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples.

3.2 Potential causes of accuracy drops

We adopt the standard classification setup and posit the existence of a "true" underlying data distribution $\mathcal{D}$ over labeled examples (x, y). The overall goal in classification is to find

[1] `https://github.com/modestyachts/CIFAR-10.1` and `https://github.com/modestyachts/ImageNetV2`

a model $\hat{f}$ that minimizes the population loss

$$L_{\mathcal{D}}(\hat{f}) \;=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\mathbb{I}[\hat{f}(x) \neq y]\Big] \,. \tag{3.2.1}$$

Since we usually do not know the distribution $\mathcal{D}$, we instead measure the performance of a trained classifier via a *test set* S_{test} drawn from the distribution $\mathcal{D}$:

$$L_{S_{\text{test}}}(\hat{f}) \;=\; \frac{1}{|S_{\text{test}}|} \sum_{(x,y)\in S_{\text{test}}} \mathbb{I}[\hat{f}(x) \neq y] \,. \tag{3.2.2}$$

We then use this test error $L_{S_{\text{test}}}(\hat{f})$ as a proxy for the population loss $L_{\mathcal{D}}(\hat{f})$. If a model $\hat{f}$ achieves a low test error, we assume that it will perform similarly well on future examples from the distribution $\mathcal{D}$. This assumption underlies essentially all empirical evaluations in machine learning since it allows us to argue that the model $\hat{f}$ generalizes.

In our experiments, we test this assumption by collecting a new test set S'_{test} from a data distribution $\mathcal{D}'$ that we carefully control to resemble the original distribution $\mathcal{D}$. Ideally, the original test accuracy $L_{S_{\text{test}}}(\hat{f})$ and new test accuracy $L_{S'_{\text{test}}}(\hat{f})$ would then match up to the random sampling error. In contrast to this idealized view, our results in Figure 3.1 show a large drop in accuracy from the original test set S_{test} set to our new test set S'_{test}. To understand this accuracy drop in more detail, we decompose the difference between $L_{S_{\text{test}}}(\hat{f})$ and $L_{S'_{\text{test}}}(\hat{f})$ into three parts (dropping the dependence on $\hat{f}$ to simplify notation):

$$L_{S_{\text{test}}} - L_{S'_{\text{test}}} \;=\; \underbrace{(L_{S_{\text{test}}} - L_{\mathcal{D}})}_{\text{Adaptivity gap}} \;+\; \underbrace{(L_{\mathcal{D}} - L_{\mathcal{D}'})}_{\text{Distribution Gap}} \;+\; \underbrace{(L_{\mathcal{D}'} - L_{S'_{\text{test}}})}_{\text{Generalization gap}} \,.$$

We now discuss to what extent each of the three terms can lead to accuracy drops.

Generalization gap. By construction, our new test set S'_{test} is independent of the existing classifier $\hat{f}$. Hence the third term $L_{\mathcal{D}'} - L_{S'_{\text{test}}}$ is the standard *generalization gap* commonly studied in machine learning. It is determined solely by the random sampling error.

A first guess is that this inherent sampling error suffices to explain the accuracy drops in Figure 3.1 (e.g., the new test set S'_{test} could have sampled certain "harder" modes of the distribution $\mathcal{D}$ more often). However, random fluctuations of this magnitude are unlikely for the size of our test sets. With 10,000 data points (as in our new ImageNet test set), a Clopper-Pearson 95% confidence interval for the test accuracy has size of at most $\pm 1\%$. Increasing the confidence level to 99.99% yields a confidence interval of size at most $\pm 2\%$. Moreover, these confidence intervals become smaller for higher accuracies, which is the relevant regime for the best-performing models. Hence random chance alone cannot explain the accuracy drops observed in our experiments.[2]

[2]We remark that the sampling process for the new test set S'_{test} could indeed *systematically* sample harder modes more often than under the original data distribution $\mathcal{D}$. Such a systematic change in the sampling process would not be an effect of random chance but captured by the distribution gap described below.

Adaptivity gap. We call the term $L_{S_{\text{test}}} - L_{\mathcal{D}}$ the *adaptivity gap*. It measures how much adapting the model $\hat{f}$ to the test set S_{test} causes the test error $L_{S_{\text{test}}}$ to underestimate the population loss $L_{\mathcal{D}}$. If we assumed that our model $\hat{f}$ is independent of the test set S_{test}, this terms would follow the same concentration laws as the generalization gap $L_{\mathcal{D}'} - L_{S'_{\text{test}}}$ above. But this assumption is undermined by the common practice of tuning model hyperparameters directly on the test set, which introduces dependencies between the model $\hat{f}$ and the test set S_{test}. In the extreme case, this can be seen as training directly on the test set. But milder forms of adaptivity may also artificially inflate accuracy scores by increasing the gap between $L_{S_{\text{test}}}$ and $L_{\mathcal{D}}$ beyond the purely random error.

Distribution gap. We call the term $L_{\mathcal{D}} - L_{\mathcal{D}'}$ the *distribution gap*. It quantifies how much the change from the original distribution $\mathcal{D}$ to our new distribution $\mathcal{D}'$ affects the model $\hat{f}$. Note that this term is not influenced by random effects but quantifies the systematic difference between sampling the original and new test sets. While we went to great lengths to minimize such systematic differences, in practice it is hard to argue whether two high-dimensional distributions are exactly the same. We typically lack a precise definition of either distribution, and collecting a real dataset involves a plethora of design choices.

3.2.1 Distinguishing between the two mechanisms

For a single model $\hat{f}$, it is unclear how to disentangle the adaptivity and distribution gaps. To gain a more nuanced understanding, we measure accuracies for *multiple* models $\hat{f}_1, \ldots, \hat{f}_k$. This provides additional insights because it allows us to determine how the two gaps have evolved over time.

For both CIFAR-10 and ImageNet, the classification models come from a long line of papers that incrementally improved accuracy scores over the past decade. A natural assumption is that later models have experienced more adaptive overfitting since they are the result of more successive hyperparameter tuning on the same test set. Their higher accuracy scores would then come from an increasing adaptivity gap and reflect progress only on the specific examples in the test set S_{test} but not on the actual distribution $\mathcal{D}$. In an extreme case, the population accuracies $L_{\mathcal{D}}(\hat{f}_i)$ would plateau (or even decrease) while the test accuracies $L_{S_{\text{test}}}(\hat{f}_i)$ would continue to grow for successive models $\hat{f}_i$.

However, this idealized scenario is in stark contrast to our results in Figure 3.1. Later models do not see diminishing returns but an *increased* advantage over earlier models. Hence we view our results as evidence that the accuracy drops mainly stem from a large distribution gap. After presenting our results in more detail in the next section, we will further discuss this point in Section 3.7.

3.3 Summary of our experiments

We now give an overview of the main steps in our reproducibility experiment. Sections 3.5 and 3.6 describe our methodology in more detail. We begin with the first decision, which was to choose informative datasets.

3.3.1 Choice of datasets

We focus on image classification since it has become the most prominent task in machine learning and underlies a broad range of applications. The cumulative progress on ImageNet is often cited as one of the main breakthroughs in computer vision and machine learning [62]. State-of-the-art models now surpass human-level accuracy by some measure [33, 81]. This makes it particularly important to check if common image classification models can reliably generalize to new data from the same source.

We decided on CIFAR-10 and ImageNet, two of the most widely-used image classification benchmarks [30]. Both datasets have been the focus of intense research for almost ten years now. Due to the competitive nature of these benchmarks, they are an excellent example for testing whether adaptivity has led to overfitting. In addition to their popularity, their carefully documented dataset creation process makes them well suited for a reproducibility experiment [53, 15, 81].

Each of the two datasets has specific features that make it especially interesting for our replication study. CIFAR-10 is small enough so that many researchers developed and tested new models for this dataset. In contrast, ImageNet requires significantly more computational resources, and experimenting with new architectures has long been out of reach for many research groups. As a result, CIFAR-10 has likely experienced more hyperparameter tuning, which may also have led to more adaptive overfitting.

On the other hand, the limited size of CIFAR-10 could also make the models more susceptible to small changes in the distribution. Since the CIFAR-10 models are only exposed to a constrained visual environment, they may be unable to learn a robust representation. In contrast, ImageNet captures a much broader variety of images: it contains about 24× more training images than CIFAR-10 and roughly 100× more pixels per image. So conventional wisdom (such as the claims of human-level performance) would suggest that ImageNet models also generalize more reliably .

As we will see, neither of these conjectures is supported by our data: CIFAR-10 models do not suffer from more adaptive overfitting, and ImageNet models do not appear to be significantly more robust.

3.3.2 Dataset creation methodology

One way to test generalization would be to evaluate existing models on new i.i.d. data from the original test distribution. For example, this would be possible if the original dataset authors had collected a larger initial dataset and randomly split it into two test sets, keeping

one of the test sets hidden for several years. Unfortunately, we are not aware of such a setup for CIFAR-10 or ImageNet.

In this thesis, we instead mimic the original distribution as closely as possible by repeating the dataset curation process that selected the original test set[3] from a larger data source. While this introduces the difficulty of disentangling the adaptivity gap from the distribution gap, it also enables us to check whether independent replication affects current accuracy scores. In spite of our efforts, we found that it is astonishingly hard to replicate the test set distributions of CIFAR-10 and ImageNet.

At a high level, creating a new test set consists of two parts:

Gathering Data. To obtain images for a new test set, a simple approach would be to use a different dataset, e.g., Open Images [52]. However, each dataset comes with specific biases [92]. For instance, CIFAR-10 and ImageNet were assembled in the late 2000s, and some classes such as `car` or `cell_phone` have changed significantly over the past decade. We avoided such biases by drawing new images from the same source as CIFAR-10 and ImageNet. For CIFAR-10, this was the larger Tiny Image dataset [93]. For ImageNet, we followed the original process of utilizing the Flickr image hosting service and only considered images uploaded in a similar time frame as for ImageNet.

In addition to the data source and the class distribution, both datasets also have rich structure *within* each class. For instance, each class in CIFAR-10 consists of images from multiple specific keywords in Tiny Images. Similarly, each class in ImageNet was assembled from the results of multiple queries to the Flickr API. We relied on the documentation of the two datasets to closely match the sub-class distribution as well.

Cleaning data. Many images in Tiny Images and the Flickr results are only weakly related to the query (or not at all). To obtain a high-quality dataset with correct labels, it is therefore necessary to manually select valid images from the candidate pool. While this step may seem trivial, our results in Section 3.4 will show that it has major impact on the model accuracies.

The authors of CIFAR-10 relied on paid student labelers to annotate their dataset. The researchers in the ImageNet project utilized Amazon Mechanical Turk (MTurk) to handle the large size of their dataset. We again replicated both annotation processes. Two graduate students impersonated the CIFAR-10 labelers, and we employed MTurk workers for our new ImageNet test set. For both datasets, we also followed the original labeling instructions, MTurk task format, etc.

After collecting a set of correctly labeled images, we sampled our final test sets from the filtered candidate pool. We decided on a test set size of 2,000 for CIFAR-10 and 10,000 for ImageNet. While these are smaller than the original test sets, the sample sizes are still large enough to obtain 95% confidence intervals of about $\pm 1\%$. Moreover, our aim was to avoid

[3]For ImageNet, we repeat the creation process of the *validation set* because most papers developed and tested models on the validation set. We discuss this point in more detail in Section 3.6.1. In the context of this thesis, we use the terms "validation set" and "test set" interchangeably for ImageNet.

bias due to CIFAR-10 and ImageNet possibly leaving only "harder" images in the respective data sources. This effect is minimized by building test sets that are small compared to the original datasets (about 3% of the overall CIFAR-10 dataset and less than 1% of the overall ImageNet dataset).

3.3.3 Results on the new test sets

CIFAR-10

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_tf	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
6	shake_shake_64d_cutout	97.1 [96.8, 97.4]	93.0 [91.8, 94.1]	4.1	5	1
16	wide_resnet_28_10	95.9 [95.5, 96.3]	89.7 [88.3, 91.0]	6.2	14	2
23	resnet_basic_110	93.5 [93.0, 93.9]	85.2 [83.5, 86.7]	8.3	24	-1
27	vgg_15_BN_64	93.0 [92.5, 93.5]	84.9 [83.2, 86.4]	8.1	27	0
30	cudaconvnet	88.5 [87.9, 89.2]	77.5 [75.7, 79.3]	11.0	30	0
31	random_features_256k_au	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12.5	31	0

ImageNet Top-1

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	72.2 [71.3, 73.1]	10.7	3	-2
4	nasnetalarge	82.5 [82.2, 82.8]	72.2 [71.3, 73.1]	10.3	1	3
21	resnet152	78.3 [77.9, 78.7]	67.0 [66.1, 67.9]	11.3	21	0
23	inception_v3_tf	78.0 [77.6, 78.3]	66.1 [65.1, 67.0]	11.9	24	-1
30	densenet161	77.1 [76.8, 77.5]	65.3 [64.4, 66.2]	11.8	30	0
43	vgg19_bn	74.2 [73.8, 74.6]	61.9 [60.9, 62.8]	12.3	44	-1
64	alexnet	56.5 [56.1, 57.0]	44.0 [43.0, 45.0]	12.5	64	0
65	fv_64k	35.1 [34.7, 35.5]	24.1 [23.2, 24.9]	11.0	65	0

Table 3.1: Model accuracies on the original CIFAR-10 test set, the original ImageNet validation set, and our new test sets. Δ Rank is the relative difference in the ranking from the original test set to the new test set in the full ordering of all models (see Section 3.10.0.3 and 3.11.0.2). For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Sections 3.10.0.2 and 3.11.0.1.

After assembling our new test sets, we evaluated a broad range of image classification models spanning a decade of machine learning research. The models include the seminal AlexNet [54], widely used convolutional networks [84, 32, 40, 89], and the state-of-the-art [13, 58]. For all deep architectures, we used code previously published online. We relied

on pre-trained models whenever possible and otherwise ran the training commands from the respective repositories. In addition, we also evaluated the best-performing approaches preceding convolutional networks on each dataset. These are random features for CIFAR-10 [75, 10] and Fisher vectors for ImageNet [71].[4] We wrote our own implementations for these models, which we also release publicly.[5]

Overall, the top-1 accuracies range from 83% to 98% on the original CIFAR-10 test set and 21% to 83% on the original ImageNet validation set. We refer the reader to Sections 3.11.0.1 and 3.10.0.2 for a full list of models and source repositories.

Figure 3.1 in the introduction plots original vs. new accuracies, and Table 3.1 in this section summarizes the numbers of key models. The remaining accuracy scores can be found in Sections 3.10.0.3 and 3.11.0.2. We now briefly describe the two main trends and discuss the results further in Section 3.7.

A significant drop in accuracy. All models see a large drop in accuracy from the original test sets to our new test sets. For widely used architectures such as VGG [84] and ResNet [32], the drop is 8% on CIFAR-10 and 11% on ImageNet. On CIFAR-10, the state of the art [13] is more robust and only drops by 3% from 98.4% to 95.5%. In contrast, the best model on ImageNet [58] sees an 11% drop from 83% to 72% in top-1 accuracy and a 6% drop from 96% to 90% in top-5 accuracy. So the top-1 drop on ImageNet is larger than what we observed on CIFAR-10.

To put these accuracy numbers into perspective, we note that the best model in the ILSVRC[6] 2013 competition achieved 89% top-5 accuracy, and the best model from ILSVRC 2014 achieved 93% top-5 accuracy. So the 6% drop in top-5 accuracy from the 2018 state-of-the-art corresponds to approximately five years of progress in a very active period of machine learning research.

Few changes in the relative order. When sorting the models in order of their original and new accuracy, there are few changes in the respective rankings. Models with comparable original accuracy tend to see a similar decrease in performance. In fact, Figure 3.1 shows that the original accuracy is highly predictive of the new accuracy and that the relationship can be summarized well with a linear function. On CIFAR-10, the new accuracy of a model is approximately given by the following formula:

$$\mathrm{acc}_{\mathrm{new}} \;=\; 1.69 \cdot \mathrm{acc}_{\mathrm{orig}} - 72.7\% \;.$$

[4]We remark that our implementation of Fisher vectors yields top-5 accuracy numbers that are 17% lower than the published numbers in ILSVRC 2012 [81]. Unfortunately, there is no publicly available reference implementation of Fisher vector models achieving this accuracy score. Hence our implementation should not be seen as an exact reproduction of the state-of-the-art Fisher vector model, but as a baseline inspired by this approach. The main goal of including Fisher vector models in our experiment is to investigate if they follow the same overall trends as convolutional neural networks.

[5]`https://github.com/modestyachts/nondeep`

[6]ILSVRC is the ImageNet Large Scale Visual Recognition Challenge [81].

On ImageNet, the top-1 accuracy of a model is given by

$$\text{acc}_{\text{new}} \;=\; 1.11 \cdot \text{acc}_{\text{orig}} - 20.2\% \;.$$

Computing a 95% confidence interval from 100,000 bootstrap samples gives $[1.63, 1.76]$ for the slope and $[-78.6, -67.5]$ for the offset on CIFAR-10, and $[1.07, 1.19]$ and $[-26.0, -17.8]$ respectively for ImageNet.

On both datasets, the slope of the linear fit is *greater* than 1. So models with higher original accuracy see a smaller drop on the new test sets. In other words, model robustness *improves* with increasing accuracy. This effect is less pronounced on ImageNet (slope 1.1) than on CIFAR-10 (slope 1.7). In contrast to a scenario with strong adaptive overfitting, neither dataset sees diminishing returns in accuracy scores when going from the original to the new test sets.

3.3.4 Experiments to test follow-up hypotheses

Since the drop from original to new accuracies is concerningly large, we investigated multiple hypotheses for explaining this drop. Sections 3.5.2 and 3.6.3 list a range of follow-up experiments we conducted, e.g., re-tuning hyperparameters, training on part of our new test set, or performing cross-validation. However, none of these effects can explain the size of the drop. We conjecture that the accuracy drops stem from small variations in the human annotation process. As we will see in the next section, the resulting changes in the test sets can significantly affect model accuracies.

3.4 Understanding the impact of data cleaning on ImageNet

A crucial aspect of ImageNet is the use of MTurk. There is a broad range of design choices for the MTurk tasks and how the resulting annotations determine the final dataset. To better understand the impact of these design choices, we assembled three different test sets for ImageNet. All of these test sets consist of images from the same Flickr candidate pool, are correctly labeled, and selected by more than 70% of the MTurk workers on average. Nevertheless, the resulting model accuracies vary by 14%. To put these numbers in context, we first describe our MTurk annotation pipeline.

MTurk Tasks. We designed our MTurk tasks and user interface to closely resemble those originally used for ImageNet. As in ImageNet, each MTurk task contained a grid of 48 candidate images for a given target class. The task description was derived from the original ImageNet instructions and included the definition of the target class with a link to a corresponding Wikipedia page. We asked the MTurk workers to select images belonging to the target class regardless of "occlusions, other objects, and clutter or text in the scene" and

to avoid drawings or paintings (both as in ImageNet). Section 3.6.1.2 shows a screenshot of our UI and a screenshot of the original UI for comparison.

For quality control, we embedded at least six randomly selected images from the original validation set in each MTurk task (three from the same class, three from a class that is nearby in the WordNet hierarchy). These images appeared in random locations of the image grid for each task. In total, we collected sufficient MTurk annotations so that we have at least 20 annotated validation images for each class.

The main outcome of the MTurk tasks is a *selection frequency* for each image, i.e., what fraction of MTurk workers selected the image in a task for its target class. We recruited at least ten MTurk workers for each task (and hence for each image), which is similar to ImageNet. Since each task contained original validation images, we could also estimate how often images from the original dataset were selected by our MTurk workers.

Sampling Strategies. In order to understand how the MTurk selection frequency affects the model accuracies, we explored three sampling strategies.

- **MatchedFrequency:** First, we estimated the selection frequency distribution for each class from the annotated original validation images. We then sampled ten images from our candidate pool for each class according to these class-specific distributions (see Section 3.6.1.4 for details).
- **Threshold0.7:** For each class, we sampled ten images with selection frequency at least 0.7.
- **TopImages:** For each class, we chose the ten images with highest selection frequency.

In order to minimize labeling errors, we manually reviewed each dataset and removed incorrect images. The average selection frequencies of the three final datasets range from 0.93 for TopImages over 0.85 for Threshold0.7 to 0.73 for MatchedFrequency. For comparison, the original validation set has an average selection frequency of 0.71 in our experiments. Hence all three of our new test sets have higher selection frequencies than the original ImageNet validation set. In the preceding sections, we presented results on MatchedFrequency for ImageNet since it is closest to the validation set in terms of selection frequencies.

Results. Table 3.2 shows that the MTurk selection frequency has significant impact on both top-1 and top-5 accuracy. In particular, TopImages has the highest average MTurk selection frequency and sees a small *increase* of about 2% in both average top-1 and top-5 accuracy compared to the original validation set. This is in stark contrast to MatchedFrequency, which has the lowest average selection frequency and exhibits a significant drop of 12% and 8%, respectively. The Threshold0.7 dataset is in the middle and sees a small decrease of 3% in top-1 and 1% in top-5 accuracy.

In total, going from TopImages to MatchedFrequency decreases the accuracies by about 14% (top-1) and 10% (top-5). For comparison, note that after excluding AlexNet (and the

Sampling Strategy	Average MTurk Selection Freq.	Average Top-1 Accuracy Change	Average Top-5 Accuracy Change
MatchedFrequency	0.73	-11.8%	-8.2%
Threshold0.7	0.85	-3.2%	-1.2%
TopImages	0.93	+2.1%	+1.8%

Table 3.2: Impact of the three sampling strategies for our ImageNet test sets. The table shows the average MTurk selection frequency in the resulting datasets and the average changes in model accuracy compared to the original validation set. We refer the reader to Section 3.4 for a description of the three sampling strategies. All three test sets have an average selection frequency of more than 0.7, yet the model accuracies still vary widely. For comparison, the original ImageNet validation set has an average selection frequency of 0.71 in our MTurk experiments. The changes in average accuracy span 14% and 10% in top-1 and top-5, respectively. This shows that details of the sampling strategy have large influence on the resulting accuracies.

SqueezeNet models tuned to match AlexNet [41]), the range of accuracies spanned by all remaining convolutional networks is roughly 14% (top-1) and 8% (top-5). So the variation in accuracy caused by the three sampling strategies is larger than the variation in accuracy among all post-AlexNet models we tested.

Figure 3.2 plots the new vs. original top-1 accuracies on Threshold0.7 and TopImages, similar to Figure 3.1 for MatchedFrequency before. For easy comparison of top-1 and top-5 accuracy plots on all three datasets, we refer the reader to Figure 3.1 in Section 3.11.0.2. All three plots show a good linear fit.

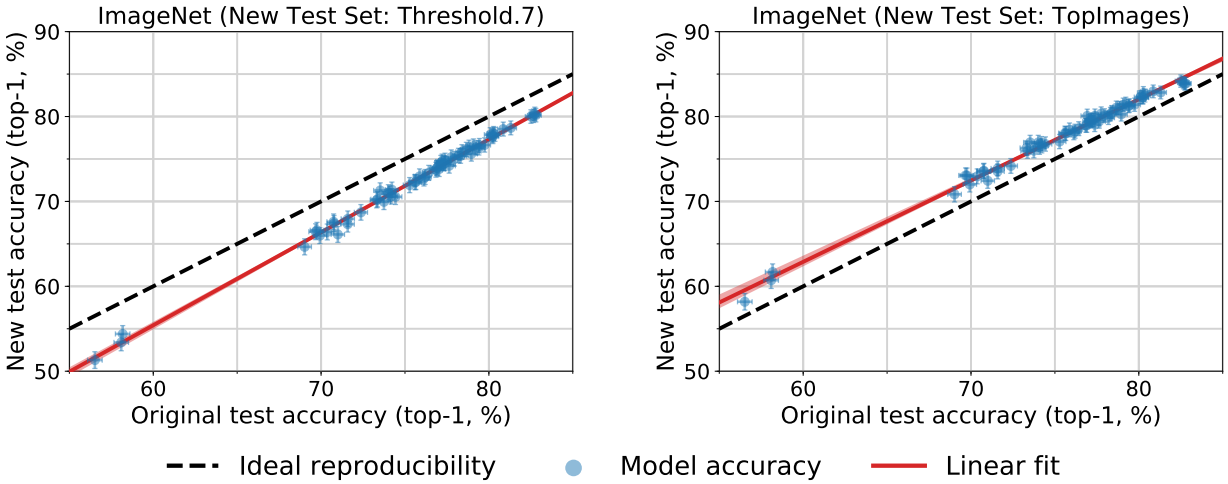


Figure 3.2: Model accuracy on the original ImageNet validation set vs. accuracy on two variants of our new test set. We refer the reader to Section 3.4 for a description of these test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). On Threshold0.7, the model accuracies are 3% lower than on the original test set. On TopImages, which contains the images most frequently selected by MTurk workers, the models perform 2% *better* than on the original test set. The accuracies on both datasets closely follow a linear function, similar to MatchedFrequency in Figure 3.1. The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples.

3.5 CIFAR-10 experiment details

We first present our reproducibility experiment for the CIFAR-10 image classification dataset [53]. There are multiple reasons why CIFAR-10 is an important example for measuring how well current models generalize to unseen data.

- CIFAR-10 is one of the most widely used datasets in machine learning and serves as a test ground for many image classification methods. A concrete measure of popularity is the fact that CIFAR-10 was the second most common dataset in NIPS 2017 (after MNIST) [30].
- The dataset creation process for CIFAR-10 is transparent and well documented [53]. Importantly, CIFAR-10 draws from the larger Tiny Images repository that has more fine-grained labels than the ten CIFAR-10 classes [93]. This enables us to minimize various forms of distribution shift between the original and new test set.
- CIFAR-10 poses a difficult enough problem so that the dataset is still the subject of active research (e.g., see [16, 28, 98, 104, 76, 13]). Moreover, there is a wide range of classification models that achieve significantly different accuracy scores. Since code for these models has been published in various open source repositories, they can be treated as independent of our new test set.

Compared to ImageNet, CIFAR-10 is significantly smaller both in the number of images and in the size of each image. This makes it easier to conduct various follow-up experiments that require training new classification models. Moreover, the smaller size of CIFAR-10 also means that the dataset has been accessible to more researchers for a longer time. Hence it is plausible that CIFAR-10 experienced more test set adaptivity than ImageNet, where it is much more costly to tune hyperparameters.

Before we describe how we created our new test set, we briefly review relevant background on CIFAR-10 and Tiny Images.

Tiny Images. The dataset contains 80 million RGB color images with resolution 32×32 pixels and was released in 2007 [93]. The images are organized by roughly 75,000 *keywords* that correspond to the non-abstract nouns from the WordNet database [65] Each keyword was entered into multiple Internet search engines to collect roughly 1,000 to 2,500 images per keyword. It is important to note that Tiny Images is a fairly noisy dataset. Many of the images filed under a certain keyword do not clearly (or not at all) correspond to the respective keyword.

CIFAR-10. The CIFAR-10 dataset was created as a cleanly labeled subset of Tiny Images for experiments with multi-layer networks. To this end, the researchers assembled a dataset consisting of ten classes with 6,000 images per class, which was published in 2009 [53]. These classes are `airplane`, `automobile`, `bird`, `cat`, `deer`, `dog`, `frog`, `horse`, `ship`, and `truck`.

The standard train / test split is class-balanced and contains 50,000 training images and 10,000 test images.

The CIFAR-10 creation process is well-documented [53]. First, the researchers assembled a set of relevant keywords for each class by using the hyponym relations in WordNet [65] (for instance, "Chihuahua" is a hyponym of "dog"). Since directly using the corresponding images from Tiny Images would not give a high quality dataset, the researchers paid student annotators to label the images from Tiny Images. The labeler instructions can be found in Section C of [53] and include a set of specific guidelines (e.g., an image should not contain two object of the corresponding class). The researchers then verified the labels of the images selected by the annotators and removed near-duplicates from the dataset via an ℓ_2 nearest neighbor search.

3.5.1 Dataset creation methodology

Our overall goal was to create a new test set that is as close as possible to being drawn from the same distribution as the original CIFAR-10 dataset. One crucial aspect here is that the CIFAR-10 dataset did not exhaust any of the Tiny Image keywords it is drawn from. So by collecting new images from the same keywords as CIFAR-10, our new test set can match the sub-class distribution of the original dataset.

Understanding the Sub-Class Distribution. As the first step, we determined the Tiny Image keyword for every image in the CIFAR-10 dataset. A simple nearest-neighbor search sufficed since every image in CIFAR-10 had an exact duplicate (ℓ_2-distance 0) in Tiny Images. Based on this information, we then assembled a list of the 25 most common keywords for each class. We decided on 25 keywords per class since the 250 total keywords make up more than 95% of CIFAR-10. Moreover, we wanted to avoid accidentally creating a harder dataset with infrequent keywords that the classifiers had little incentive to learn based on the original CIFAR-10 dataset.

The keyword distribution can be found in Section 3.10.0.1. Inspecting this list reveals the importance of matching the sub-class distribution. For instance, the most common keyword in the `airplane` class is `stealth_bomber` and not a more common civilian type of airplane. In addition, the third most common keyword for the `airplane` class is `stealth_fighter`. Both types of planes are highly distinctive. There are more examples where certain sub-classes are considerably different. For instance, trucks from the keyword `fire_truck` are mostly red, which is quite different from pictures for `dump_truck` or other keywords.

Collecting new images. After determining the keywords, we collected corresponding images. To simulate the student / researcher split in the original CIFAR-10 collection procedure, we introduced a similar split among two authors of this paper. Author A took the role of the original student annotators and selected new suitable images for the 250 keywords. In order to ensure a close match between the original and new images for each keyword, we

built a user interface that allowed Author A to first look through existing CIFAR-10 images for a given keyword and then select new candidates from the remaining pictures in Tiny Images. Author A followed the labeling guidelines in the original instruction sheet [53]. The number of images Author A selected per keyword was so that our final dataset would contain between 2,000 and 4,000 images. We decided on 2,000 images as a target number for two reasons:

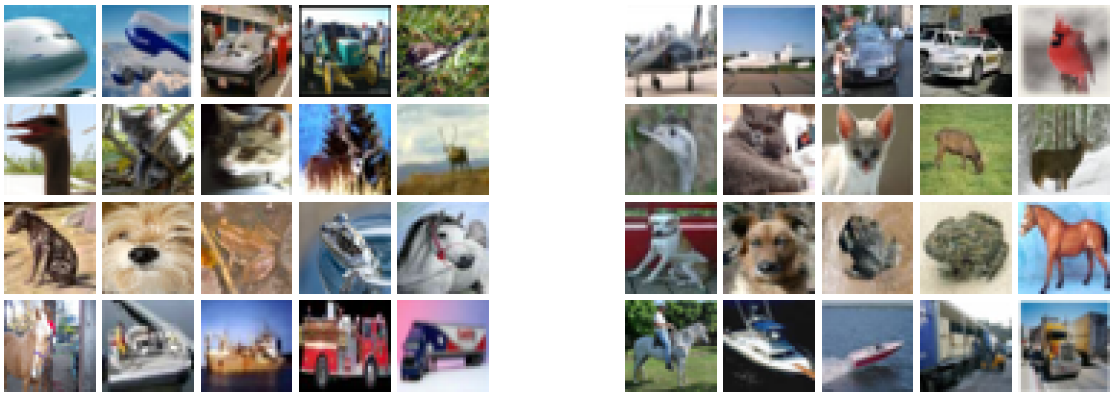
- While the original CIFAR-10 test set contains 10,000 images, a test set of size 2,000 is already sufficient for a fairly small confidence interval. In particular, a conservative confidence interval (Clopper-Pearson at confidence level 95%) for accuracy 90% has size about $\pm 1\%$ with $n = 2{,}000$ (to be precise, [88.6%, 91.3%]). Since we considered a potential discrepancy between original and new test accuracy only interesting if it is significantly larger than 1%, we decided that a new test set of size 2,000 was large enough for our study.

- As with very infrequent keywords, our goal was to avoid accidentally creating a harder test set. Since some of the Tiny Image keywords have only a limited supply of remaining adequate images, we decided that a smaller target size for the new dataset would reduce bias to include images of more questionable difficulty.

After Author A had selected a set of about 9,000 candidate images, Author B adopted the role of the researchers in the original CIFAR-10 dataset creation process. In particular, Author B reviewed all candidate images and removed images that were unclear to Author B or did not conform to the labeling instructions in their opinion (some of the criteria are subjective). In the process, a small number of keywords did not have enough images remaining to reach the $n = 2{,}000$ threshold. Author B then notified Author A about the respective keywords and Author A selected a further set of images for these keywords. In this process, there was only one keyword where Author A had to carefully examine all available images in Tiny Images. This keyword was `alley_cat` and comprises less than 0.3% of the overall CIFAR-10 dataset.

Final assembly. After collecting a sufficient number of high-quality images for each keyword, we sampled a random subset from our pruned candidate set. The sampling procedure was such that the keyword-level distribution of our new dataset matches the keyword-level distribution of CIFAR-10 (see Section 3.10.0.1). In the final stage, we again proceeded similar to the original CIFAR-10 dataset creation process and used ℓ_2-nearest neighbors to filter out near duplicates. In particular, we removed near-duplicates within our new dataset and also images that had a near duplicate in the original CIFAR-10 dataset (train or test). The latter aspect is particularly important since our reproducibility study is only interesting if we evaluate on truly unseen data. Hence we manually reviewed the top-10 nearest neighbors for each image in our new test set. After removing near-duplicates in our dataset, we re-sampled the respective keywords until this process converged to our final dataset.

Figure 3.3b shows a random subset of images from the original and our new test set.



(a) Test set A (b) Test set B

Figure 3.3: Randomly selected images from the original and new CIFAR-10 test sets. Each grid contains two images for each of the ten classes. The following footnote reveals which of the two grids corresponds to the new test set.[7]

We remark that we did not run any classifiers on our new dataset during the data collection phase of our study. In order to ensure that the new data does not depend on the existing classifiers, it is important to strictly separate the data collection phase from the following evaluation phase.

3.5.2 Follow-up hypotheses

Since the gap between original and new accuracy is concerningly large, we investigated multiple hypotheses for explaining this gap.

3.5.2.1 Statistical Error

A first natural guess is that the gap is simply due to statistical fluctuations. But as noted before, the sample size of our new test set is large enough so that a 95% confidence interval has size about $\pm 1.2\%$. Since a 95% confidence interval for the original CIFAR-10 test accuracy is even smaller (roughly $\pm 0.6\%$ for 90% classification accuracy and $\pm 0.3\%$ for 97% classification accuracy), we can rule out statistical error as the main explanation.

3.5.2.2 Differences in Near-Duplicate Removal

As mentioned in Section 3.5.1, the final step of both the original CIFAR-10 and our dataset creation procedure is to remove near-duplicates. While removing near-duplicates between our new test set and the original CIFAR-10 dataset, we noticed that the original test set contained images that we would have ruled out as near-duplicates. A large number of near-duplicates between CIFAR-10 train and test, combined with our more stringent near-duplicate removal, could explain some of the accuracy drop. Indeed, we found about 800

images in the original CIFAR-10 test set that we would classify as near-duplicates (8% of the entire test set). Moreover, most classifiers have accuracy between 99% and 100% on these near-duplicates (recall that most models achieve 100% training error). However, the following calculation shows that the near-duplicates can explain at most 1% of the observed difference.

For concreteness, we consider a model with 93% original test set accuracy such as a common VGG or ResNet architecture. Let acc_{true} be the "true" accuracy of the model on test images that are not near-duplicates, and let acc_{nd} be the accuracy on near-duplicates. Then for 8% near-duplicates, the overall accuracy is given by

$$\text{acc} \;=\; 0.92 \cdot \text{acc}_{\text{true}} + 0.08 \cdot \text{acc}_{\text{nd}} \;.$$

Using $\text{acc} = 0.93$, $\text{acc}_{\text{nd}} = 1.0$, and solving for acc_{true} then yields $\text{acc}_{\text{true}} \approx 0.924$. So the accuracy on original test images that are not near-duplicates is indeed lower, but only by a small amount (0.6%). This is in contrast to the 8% - 9% accuracy drop that VGG and ResNet models with 93% original accuracy see in our experiments.

For completeness, we describe our process for finding near duplicates in detail. For every test image, we visually inspected the top-10 nearest neighbors in both ℓ_2-distance and the SSIM (structural similarity) metric. We compared the original test set to the CIFAR-10 training set, and our new test set to both the original training and test sets. We consider an image pair as near-duplicates if both images have the same object in the same pose. We include images that have different zoom, color scale, stretch in the horizontal or vertical direction, or small shifts in vertical or horizontal position. If the object was rotated or in a different pose, we did not include it as a near-duplicate.

3.5.2.3 Hyperparameter Tuning

Another conjecture is that we can recover some of the missing accuracy by re-tuning hyperparameters of a model. To this end, we performed a grid search over multiple parameters of a VGG model. We selected three standard hyperparameters known to strongly influence test set performance: initial learning rate, dropout, and weight decay. The `vgg16_keras` architecture uses different amounts of dropout across different layers of the network, so we chose to tune a multiplicative scaling factor for the amount of dropout. This keeps the ratio of dropout across different layers constant.

We initialized a hyperparameter configuration from values tuned to the original test set (learning rate 0.1, dropout ratio 1, weight decay 5×10^{-4}), and performed a grid search across the following values:

- Learning rate in $\{0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\}$.
- Dropout ratio in $\{0.5, 0.75, 1, 1.25, 1.75\}$.

[7] Test Set A is the new test set and Test Set B is the original test set.

- Weight decay in $\{5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}\}$.

We ensured that the best performance was never at an extreme point of any range we tested for an individual hyperparameter. Overall, we did not find a hyperparameter setting with a significantly better accuracy on the new test set (the biggest improvement was from 85.3% to 85.8%).

3.5.2.4 Visually Inspecting Hard Images

It is also possible that we accidentally created a more difficult test set by including a set of "harder" images. To explore this question, we visually inspected the set of images that most models incorrectly classified. Figure 3.11 in Section 3.10.0.5 shows examples of the hard images in our new test set that no model correctly classified. We find that all the new images are valid images that are recognizable to humans.

3.5.2.5 Human Accuracy Comparison

The visual inspection of hard images in the previous section is one way to compare the original and new test sets. However, our conclusion may be biased since we have created the new test set ourselves. To compare the relative hardness of the two test sets more objectively, we also conducted a small experiment to measure human accuracy on the two test sets.[8] The goal of the experiment was to measure if human accuracy is significantly different on the original and new test sets.

Since we conjectured that our new test set included particularly hard images, we focused our experiment on the approximately 5% hardest images in both test sets. Here, "hardness" is defined by how many models correctly classified an image. After rounding to include all images that were classified by the same number of models, we obtained 500 images from the original test set and 115 images from our new test set.

We recruited nine graduate students from three different research groups in the Electrical Engineering & Computer Sciences Department at UC Berkeley. We wrote a simple user interface that allowed the participants to label images with one of the ten CIFAR-10 classes. To ensure that the participants did not know which dataset an image came from, we presented the images in random order.

Table 3.3 shows the results of our experiment. We find that four participants performed better on the original test set and five participants were better on our new test set. The average difference is -0.8%, i.e., the participants do not see a drop in average accuracy on this subset of original and new test images. This suggests that our new test set is not significantly harder for humans. However, we remark that our results here should only be seen as a preliminary study. Understanding human accuracy on CIFAR-10 in more detail will require further experiments.

[8]Use of this data was permitted by the Berkeley Committee for Protection of Human Subjects (CPHS).

	Human Accuracy (%)		
	Original Test Set	New Test Set	Gap
Participant 1	85 [81.6, 88.0]	83 [74.2, 89.8]	2
Participant 2	83 [79.4, 86.2]	81 [71.9, 88.2]	2
Participant 3	82 [78.3, 85.3]	78 [68.6, 85.7]	4
Participant 4	79 [75.2, 82.5]	84 [75.3, 90.6]	-5
Participant 5	76 [72.0, 79.7]	77 [67.5, 84.8]	-1
Participant 6	75 [71.0, 78.7]	73 [63.2, 81.4]	2
Participant 7	74 [69.9, 77.8]	79 [69.7, 86.5]	-5
Participant 8	74 [69.9, 77.8]	76 [66.4, 84.0]	-2
Participant 9	67 [62.7, 71.1]	71 [61.1, 79.6]	-4

Table 3.3: Human accuracy on the "hardest" images in the original and our new CIFAR-10 test set. We ordered the images by number of incorrect classifications from models in our testbed and then selected the top 5% images from the original and new test set (500 images from the original test set, 115 images from our new test set). The results show that on average humans do not see a drop in accuracy on this subset of images.

3.5.2.6 Training on Part of Our New Test Set

If our new test set distribution is significantly different from the original CIFAR-10 distribution, retraining on part of our new test set (plus the original training data) may improve the accuracy on the held-out fraction of our new test set.

We conducted this experiment by randomly drawing a class-balanced split containing about 1,000 images from the new test set. We then added these images to the full CIFAR-10 training set and retrained the `vgg16_keras` model. After training, we tested the model on the remaining half of the new test set. We repeated this experiment twice with different randomly selected splits from our test set, obtaining accuracies of 85.1% and 85.4% (compared to 84.9% without the extra training data[9]). This provides evidence that there is no large distribution shift between our new test set and the original CIFAR-10 dataset, or that the model is unable to learn the modified distribution.

3.5.2.7 Cross-validation

Cross-validation can be a more reliable way of measuring a model's generalization ability than using only a single train / test split. Hence we tested if cross-validation on the original CIFAR-10 dataset could predict a model's error on our new test set. We created cross-validation data by randomly dividing the training set into 5 class-balanced splits. We then

[9]This number is slightly lower than the accuracy of `vgg16_keras` on our new test set in Table 3.12, but still within the 95% confidence interval [83.6, 86.8]. Hence we conjecture that the difference is due to the random fluctuation arising from randomly initializing the model.

randomly shuffled together 4 out of the 5 training splits with the original test set. The leftover held-out split from the training set then became the new test set.

We retrained the models `vgg_15_BN_64`, `wide_resnet_28_10`, and `shake_shake_64d_cutout` on each of the 5 new datasets we created. The accuracies are reported in Table 3.4. The accuracies on the cross-validation splits did not differ much from the accuracy on the original test set. The variation among the cross-validation splits is significantly smaller than the drop on our new test set.

	Model Accuracy (%)		
Dataset	`vgg_15_BN_64`	`wide_resnet_28_10`	`shake_shake_64d_cutout`
Original Test Set	93.6 [93.1, 94.1]	95.7 [95.3, 96.1]	97.1 [96.8, 97.4]
Split 1	93.9 [93.4, 94.3]	96.2 [95.8, 96.6]	97.2 [96.9, 97.5]
Split 2	93.8 [93.3, 94.3]	96.0 [95.6, 96.4]	97.3 [97.0, 97.6]
Split 3	94.0 [93.5, 94.5]	96.4 [96.0, 96.8]	97.4 [97.1, 97.7]
Split 4	94.0 [93.5, 94.5]	96.2 [95.8, 96.6]	97.4 [97.1, 97.7]
Split 5	93.5 [93.0, 94.0]	96.5 [96.1, 96.9]	97.4 [97.1, 97.7]
New Test Set	84.9 [83.2, 86.4]	89.7 [88.3, 91.0]	93.0 [91.8, 94.1]

Table 3.4: Model accuracies on cross-validation splits for the original CIFAR-10 data. The difference in cross-validation accuracies is significantly smaller than the drop to the new test set.

3.5.2.8 Training a Discriminator for Original vs. New Test Set

Our main hypothesis for the accuracy drop is that small variations in the test set creation process suffice to significantly reduce a model's accuracy. To test whether these variations could be detected by a convolutional network, we investigated whether a discriminator model could distinguish between the two test sets.

We first created a training set consisting of $3,200$ images (1,600 from the original test set and 1,600 from our new test set) and a test set of 800 images (consisting of 400 images from original and new test set each). Each image had a binary label indicating whether it came from the original or new test set. Additionally, we ensured that that both datasets were class balanced.

We then trained `resnet_32` and `resnet_110` models for 160 epochs using a standard SGD optimizer to learn a binary classifier between the two datasets. We conducted two variants of this experiment: in one variant, we trained the model from scratch. In the other variant, we started with a model pre-trained on the regular CIFAR-10 classification task.

Our results are summarized in Table 3.5. Overall we found that the resulting models could not discriminate well between the original and our new test set: the best accuracy we obtained is 53.1%.

Model	Discriminator Accuracy (%) random initialization	Discriminator Accuracy (%) pre-trained
`resnet_32`	50.1 [46.6, 53.6]	52.9 [49.4, 56.4]
`resnet_110`	50.3 [46.7, 53.8]	53.1 [49.6, 56.6]

Table 3.5: Accuracies for discriminator models trained to distinguish between the original and new CIFAR-10 test sets. The models were initialized either randomly or using a model pre-trained on the original CIFAR-10 dataset. Although the models performed slightly better than random chance, the confidence intervals (95% Clopper Pearson) still overlap with 50% accuracy.

3.5.2.9 An Exactly Class-balanced Test Set

The top 25 keywords of each class in CIFAR-10 capture approximately 95% of the dataset. However, the remaining 5% of the dataset are skewed towards the class `ship`. As a result, our new dataset was not exactly class-balanced and contained only 8% images of class `ship` (as opposed to 10% in the original test set).

To measure whether this imbalance affected the accuracy scores, we created an exactly class-balanced version of our new test set with 2,000 images (200 per class). In this version, we selected the top 50 keywords in each class and computed a fractional number of images for each keyword. We then rounded these numbers so that images for keywords with the largest fractional part were added first. The resulting model accuracies can be found in Table 3.13 (Section 3.10.0.4). Models with lower original accuracies achieve a small accuracy improvement on the exactly class-balanced test set (around 0.3%), but the accuracy drop of the best-performing model remains unchanged.

3.6 ImageNet experiment details

Our results on CIFAR-10 show that current models fail to reliably generalize in the presence of small variations in the data distribution. One hypothesis is that the accuracy drop stems from the limited nature of the CIFAR-10 dataset. Compared to other datasets, CIFAR-10 is relatively small, both in terms of image resolution and the number of images in the dataset. Since the CIFAR-10 models are only exposed to a constrained visual environment, they may be unable to learn a more reliable representation.

To investigate whether ImageNet models generalize more reliably, we assemble a new test set for ImageNet. ImageNet captures a much broader variety of natural images: it contains about 24× more training images than CIFAR-10 with roughly 100× more pixels per image. As a result, ImageNet poses a significantly harder problem and is among the most prestigious machine learning benchmarks. The steadily improving accuracy numbers have also been cited as an important breakthrough in machine learning [62]. If popular ImageNet models are indeed more robust to natural variations in the data (and there is again no adaptive overfitting), the accuracies on our new test set should roughly match the existing accuracies.

Before we proceed to our experiments, we briefly describe the relevant background concerning the ImageNet dataset. For more details, we refer the reader to the original ImageNet publications [15, 81].

ImageNet. ImageNet [15, 81] is a large image database consisting of more than 14 million human-annotated images depicting almost 22,000 classes. The images do not have a uniform size, but most of them are stored as RGB color images with a resolution around 500×400 pixels. The classes are derived from the WordNet hierarchy [65], which represents each class by a set of synonyms ("synset") and is organized into semantically meaningful relations. Each class has an associated definition ("gloss") and a unique WordNet ID ("wnid").

The ImageNet team populated the classes with images downloaded from various image search engines, using the WordNet synonyms as queries. The researchers then annotated the images via Amazon Mechanical Turk (MTurk). A class-specific threshold decided how many agreements among the MTurk workers were necessary for an image to be considered valid. Overall, the researchers employed over 49,000 workers from 167 countries [55].

Since 2010, the ImageNet team has run the yearly ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which consists of separate tracks for object classification, localization, and detection. All three tracks are based on subsets of the ImageNet data. The classification track has received the most attention and is also the focus of this chapter.

The ILSVRC2012 competition data has become the de facto benchmark version of the dataset and comprises 1.2 million training images, 50,000 validation images, and 100,000 test images depicting 1,000 categories. We generally refer to this data as the ImageNet training, validation, and test set. The labels for the ImageNet test set were never publicly released in order to minimize adaptive overfitting. Instead, teams could submit a limited number of requests to an evaluation server in order to obtain accuracy scores. There were no similar

limitations in place for the validation set. Most publications report accuracy numbers on the validation set.

The training, validation, and test sets were not drawn strictly i.i.d. from the same distribution (i.e., there was not a single data collection run with the result split randomly into training, validation, and test). Instead, the data collection was an ongoing process and both the validation and test sets were refreshed in various years of the ILSVRC. One notable difference is that the ImageNet training and validation sets do not have the same data source: while the ImageNet training set consists of images from several search engines (e.g., Google, MSN, Yahoo, and Flickr), the validation set consists almost entirely of images from Flickr [1].

3.6.1 Dataset creation methodology

Since the existing training, validation, and test sets are not strictly i.i.d. (see above), the first question was which dataset part to replicate. For our experiment, we decided to match the distribution of the *validation set*. There are multiple reasons for this choice:

- In contrast to the training set, the validation set comes from only one data source (Flickr). Moreover, the Flickr API allows fine-grained searches, which makes it easier to control the data source and match the original distribution.
- In contrast to the original test set, the validation set comes with label information. This makes it easier to inspect the existing image distribution for each class, which is important to ensure that we match various intricacies of the dataset (e.g., see Section 3.11.0.6 for examples of ambiguous classes).
- Most papers report accuracy numbers on the validation set. Hence comparing new vs. existing accuracies is most relevant for the validation set.
- The validation set is commonly used to develop new architectures and tune hyperparameters, which leads to the possibility of adaptive overfitting. If we again observe no diminishing returns in accuracy on our new test set, this indicates that even the validation set is resilient to adaptive overfitting.

Therefore, our goal was to replicate the distribution of the original validation set as closely as possible. We aimed for a new test set of size 10,000 since this would already result in accuracy scores with small confidence intervals (see Section 3.2). While a larger dataset would result in even smaller confidence intervals, we were also concerned that searching for more images might lead to a larger distribution shift. In particular, we decided to use a time range for our Flickr queries *after* the original ImageNet collection period (see below for the corresponding considerations). Since a given time period only has a limited supply of high quality images, a larger test set would have required a longer time range. This in turn may create a larger temporal distribution shift. To balance these two concerns, we decided on a size of 10,000 images for the new test set.

Figure 3.4 presents a visual overview of our dataset creation pipeline. It consists of two parts: creating a pool of candidate images and sampling a clean dataset from this candidate pool. We now describe each part in detail to give the reader insights into the design choices potentially affecting the final distribution.

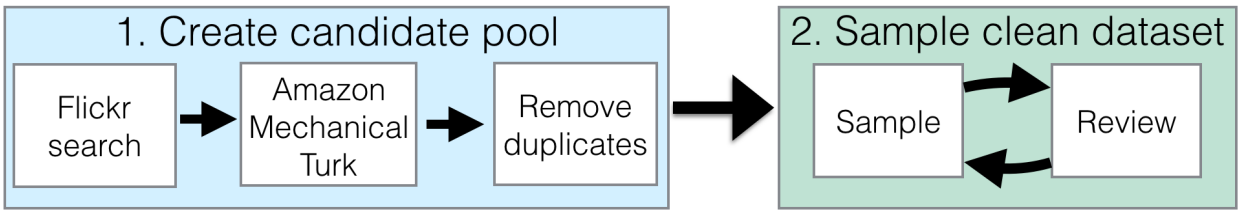


Figure 3.4: The pipeline for creating the new ImageNet test set. It consists of two parts: creating the candidate pool and sampling the final dataset from this candidate pool.

3.6.1.1 Creating a Candidate Pool

Similar to the creation procedure for the original ImageNet validation set, we collected candidate images from the Flickr image hosting service and then annotated them with Amazon Mechanical Turk (MTurk).

Downloading images from Flickr. The Flickr API has a range of parameters for image searches such as the query terms, an allowed time range, a maximum number of returned images, and a sorting order. We summarize the main points here:

- **Query terms:** For each class, we used each of the WordNet synonyms as a search term in separate queries.
- **Date range:** There were two main options for the date range associated with our queries to Flickr: either the same date range as the original ImageNet data collection, or a date range directly after ImageNet. The advantage of using the ImageNet date range is that it avoids a distribution shift due to the time the images were taken. However, this option also comes with two important caveats: First, the pool of high quality images in the original ImageNet date range could have been largely exhausted by ImageNet. Second, the new dataset could end up with near-duplicates of images in the original validation or training set that are hard to detect. Especially the first issue is difficult to quantify, so we decided on a time range directly after the ImageNet collection period.

  In particular, we initially searched for images taken and uploaded to Flickr between July 11, 2012 and July 11, 2013 because the final ILSVRC2012 public data release was on July 10, 2012. Since we used a period of only one year (significantly shorter than the ImageNet collection period), we believe that the temporal component of the distribution shift is small.

- **Result size:** We initially downloaded up to 100 images for each class. If a class has k synonyms associated with it, we requested $100/k$ images for each synonym. We decided on 100 images per class since we aimed for 10,000 images overall and estimated that 10% of the candidate images would be of sufficiently high quality (similar to ImageNet [15]).
- **Result order:** Flickr offers the sorting options "relevance", "interestingness", and various temporal orderings. Note that the "relevance" and "interestingness" orderings may rely on machine learning models trained on ImageNet. Since these orderings may introduce a significant bias (e.g., by mainly showing images that current ImageNet models recognize for the respective search term), we chose to order the images by their upload date. This helps to ensure that our new test set is independent of current classifiers.

After our first data collection, we found it necessary to expand the initial candidate pool for particular classes in order to reach a sufficient number of valid images. This is similar to the original ImageNet creation process, where the authors expanded the set of queries using two methods [15, 81]. The first method appended a word from the parent class to the queries if this word also appeared in the gloss of the target class. The second method included translations of the queries into other languages such as Chinese, Spanish, Dutch, and Italian.

We took the following steps to expand our search queries, only proceeding to the next step for a given class when in need of more images.

1. Append a word from the parent class if the word appears in the gloss of the target class.
2. Expand the maximum number of images to 200 for this class.
3. Expand the search range to include photos taken or uploaded before July 11, 2014 (i.e., a time span of two years instead of one).
4. Concatenate compound queries, i.e., search for "dialphone" instead of "dial phone".
5. Manually pick alternative query words, including translations of the queries.

In total, we obtained 208,145 candidate images from Flickr.

Amazon Mechanical Turk. While the candidate images from Flickr are correlated with their corresponding class, a large number of images are still unsuitable for an image classification dataset. For instance, the images may be of low quality (blurry, unclear object presence, etc.), violate dataset rules (e.g., no paintings), or be simply unrelated to the target class. So similar to ImageNet, we utilized MTurk to filter our pool of candidate images.

We designed our MTurk tasks and UI to be close to those used in ImageNet. As in ImageNet, we showed each MTurk worker a grid of 48 candidate images for a given target

class. The task description was derived from the original ImageNet instructions and included the definition of the target class with a link to a corresponding Wikipedia page. We asked the MTurk workers to select images belonging to the target class regardless of "occlusions, other objects, and clutter or text in the scene" and to avoid drawings or paintings (both as in ImageNet). Section 3.6.1.2 shows a screenshot of our UI and a screenshot of the original UI for comparison.

For quality control, we embedded at least six randomly selected images from the original validation set in each MTurk task (three from the same class, three from a class that is nearby in the WordNet hierarchy). These images appeared in random locations of the image grid for each task. We obfuscated all image URLs and resized our images to match the most common size of the existing validation images so that the original validation images were not easy to spot.

The main outcome of the MTurk tasks is a *selection frequency* for each image, i.e., what fraction of MTurk workers selected the image in a task for its target class. We recruited at least ten MTurk workers for each task (and hence for each image), which is similar to ImageNet. Since each task contained original validation images, we could also estimate how often images from the original dataset were selected by our MTurk workers.

Removing near-duplicate images. The final step in creating the candidate pool was to remove near-duplicates, both within our new test set and between our new test set and the original ImageNet dataset. Both types of near-duplicates could harm the quality of our dataset.

Since we obtained results from Flickr in a temporal ordering, certain events (e.g., the 2012 Olympics) led to a large number of similar images depicting the same scene (e.g., in the class for the "horizontal bar" gymnastics instrument). Inspecting the ImageNet validation set revealed only very few sets of images from a single event. Moreover, the ImageNet paper also remarks that they removed near-duplicates [15]. Hence we decided to remove near-duplicates within our new test set.

Near-duplicates between our dataset and the original test set are also problematic. Since the models typically achieve high accuracy on the training set, testing on a near-duplicate of a training image checks for memorization more than generalization. A near-duplicate between the existing validation set and our new test set also defeats the purpose of measuring generalization to previously unseen data (as opposed to data that may already have been the victim of adaptive overfitting).

To find near-duplicates, we computed the 30 nearest neighbors for each candidate image in three different metrics: ℓ_2-distance on raw pixels, ℓ_2-distance on features extracted from a pre-trained VGG [84] model (fc7), and SSIM (structural similarity) [94], which is a popular image similarity metric. For metrics that were cheap to evaluate (ℓ_2-distance on pixels and ℓ_2-distance on fc7), we computed nearest neighbor distances to all candidate images and all of the original ImageNet data. For the more compute-intensive SSIM metric, we restricted the set of reference images to include all candidate images and the five closest ImageNet

classes based on the tree distance in the WordNet hierarchy. We then manually reviewed nearest neighbor pairs below certain thresholds for each metric and removed any duplicates we discovered.

To the best of our knowledge, ImageNet used only nearest neighbors in the ℓ_2-distance to find near-duplicates [1]. While this difference may lead to a small change in distribution, we still decided to use multiple metrics since including images that have near-duplicates in ImageNet would be contrary to the main goal of our experiment. Moreover, a manual inspection of the original validation set revealed only a very small number of near-duplicates within the existing dataset.

3.6.1.2 MTurk user interfaces

For comparison, we include the original ImageNet MTurk user interface (UI) in Figure 3.5 and the MTurk UI we used in our experiments in Figure 3.6. Each UI corresponds to one task for the MTurk workers, which consists of 48 images in both cases. In contrast to the original ImageNet UI, our UI takes up more than one screen. This requires the MTurk workers to scroll but also provides more details in the images. While the task descriptions are not exactly the same, they are very similar and contain the same directions (e.g., both descriptions ask the MTurk workers to exclude drawings or paintings).

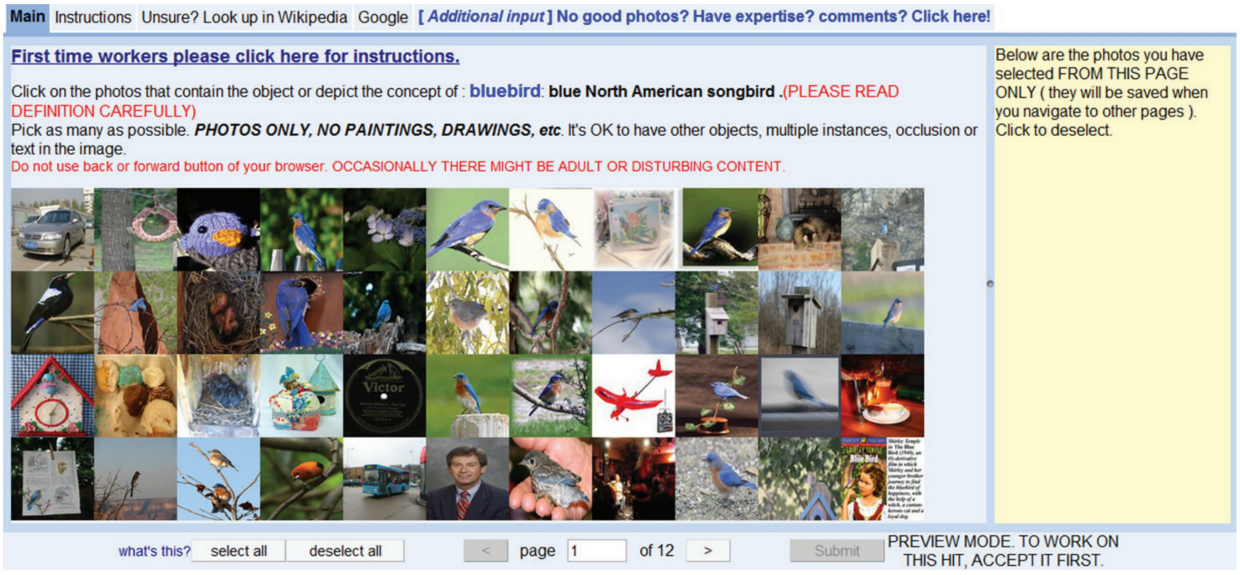


Figure 3.5: The user interface employed in the original ImageNet collection process for the labeling tasks on Amazon Mechanical Turk.

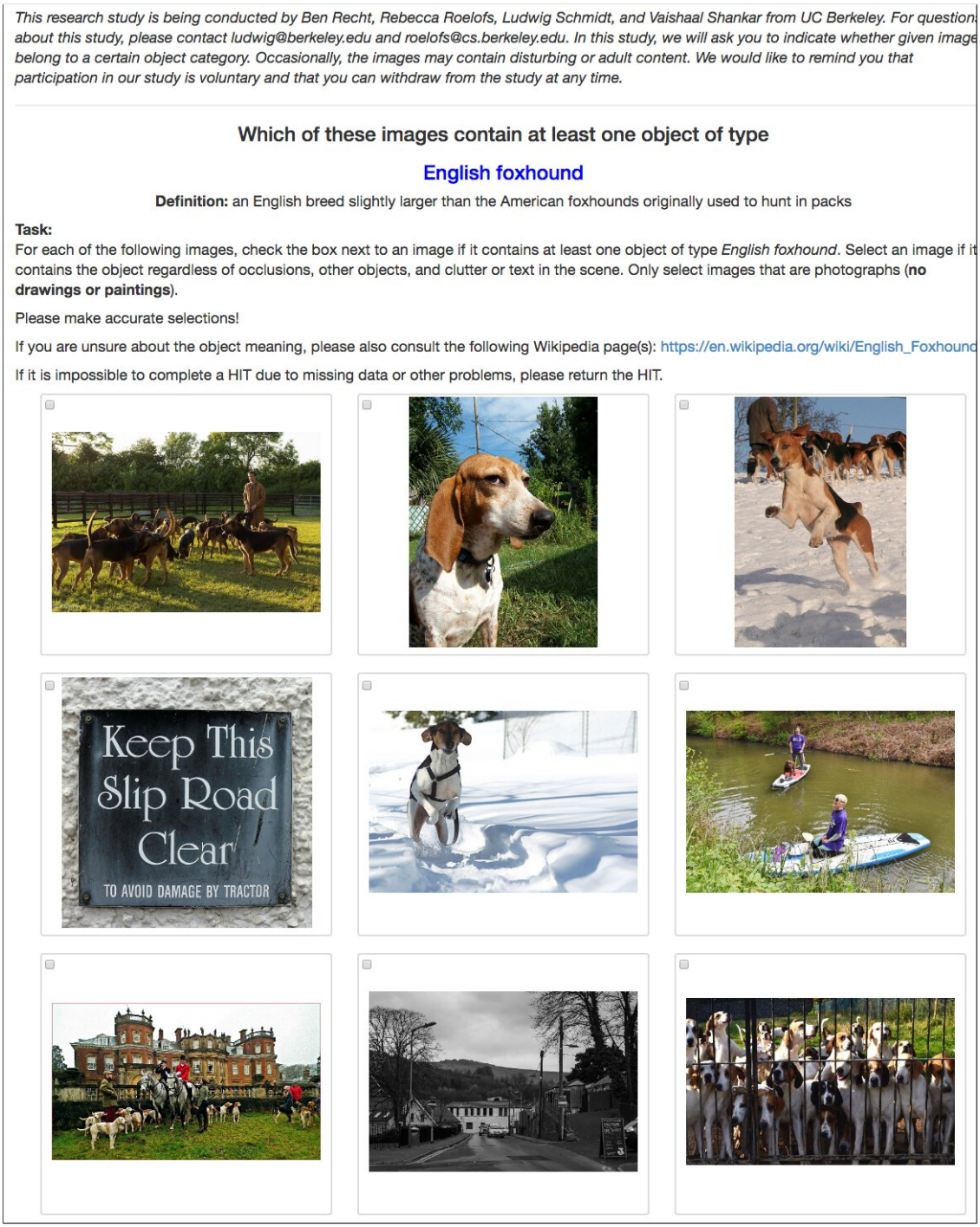


Figure 3.6: Our user interface for labeling tasks on Amazon Mechanical Turk. The screenshot here omits the scroll bar and shows only a subset of the entire MTurk task. As in the ImageNet UI, the full task involves a grid of 48 images.

3.6.1.3 User interface for our final reviewing process

Figure 3.7 shows a screenshot of the reviewing UI that we used to manually review the new ImageNet datasets. At the top, the UI displays the wnid ("n01667114"), the synset (**mud turtle**), and the gloss. Next, a grid of 20 images is shown in 4 rows.

The top two rows correspond to the candidate images currently sampled for the new dataset. Below each image, our UI shows a unique identifier for the image and the date the image was taken. There is also a check box to blacklist any incorrect images. In addition, there is a check box for each image to add it to the blacklist of incorrect images. If an image is added to the blacklist, it will be removed in the next revision of the dataset and replaced with a new image from the candidate pools. The candidate images are sorted by the date they were taken, which makes it easier to spot and remove near-duplicates. Images are marked as near-duplicates by adding their identifier to the "Near-duplicate set" text field.

The bottom two rows correspond to a random sample of images from the validation set that belong to the target class. We display these images to make it easier to detect and correct for distribution shifts between our new test sets and the original ImageNet validation dataset.

3.6.1.4 Sampling a Clean Dataset

The result of collecting a candidate pool was a set of images with annotations from MTurk, most importantly the selection frequency of each image. In the next step, we used this candidate pool to sample a new test set that closely resembles the distribution of the existing validation set. There were two main difficulties in this process.

First, the ImageNet publications do not provide the agreement thresholds for each class that were used to determine which images were valid. One possibility could be to run the algorithm the ImageNet authors designed to compute the agreement thresholds. However, this algorithm would need to be exactly specified, which is unfortunately not the case to the best of our knowledge.[10]

Second, and more fundamentally, it is impossible to exactly replicate the MTurk worker population from 2010 – 2012 with a reproducibility experiment in 2018. Even if we had access to the original agreement thresholds, it is unclear if they would be meaningful for our MTurk data collection (e.g., because the judgments of our annotations could be different). Similarly, re-running the algorithm for computing agreement thresholds could give different results with our MTurk worker population.

[10]To be precise: Jia Deng's PhD thesis [14] provides a clear high-level description of their algorithm for computing agreement thresholds. However – as is commonly the case in synopses of algorithms – the description still omits some details such as the binning procedure or the number of images used to compute the thresholds. Since it is usually hard to exactly reconstruct a non-trivial algorithm from an informal summary, we instead decided to implement three different sampling strategies and compare their outcomes. Potential deviations from the ImageNet sampling procedure are also alleviated by the fact that our MTurk tasks always included at least a few images from the original validation set, which allowed us to calibrate our sampling strategies to match the existing ImageNet data.

So instead of attempting to directly replicate the original agreement thresholds, we instead explored three different sampling strategies. An important asset in this part of our experiment was that we had inserted original validation images into the MTurk tasks (see the previous subsection). So at least for *our* MTurk worker population, we could estimate how frequently the MTurk workers select the original validation images.

In this subsection, we describe our sampling strategy that closely matches the selection frequency distribution of the original validation set. The follow-up experiments in Section 3.4 then explore the impact of this design choice in more detail. As we will see, the sampling strategy has significant influence on the model accuracies.

Matching the Per-class Selection Frequency. A simple approach to matching the selection frequency of the existing validation set would be to sample new images so that the mean selection frequency is the same as for the original dataset. However, a closer inspection of the selection frequencies reveals significant differences between the various classes. For instance, well-defined and well-known classes such as "African elephant" tend to have high selection frequencies ranging from 0.8 to 1.0. At the other end of the spectrum are classes with an unclear definition or easily confused alternative classes. For instance, the MTurk workers in our experiment often confused the class "nail" (the fastener) with fingernails, which led to significantly lower selection frequencies for the original validation images belonging to this class. In order to match these class-level details, we designed a sampling process that approximately matches the selection frequency distribution for each class.

As a first step, we built an estimate of the per-class distribution of selection frequencies. For each class, we divided the annotated validation images into five histogram bins based on their selection frequency. These frequency bins were $[0.0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, and $[0.8, 1.0]$. Intuitively, these bins correspond to a notion of image quality assessed by the MTurk workers, with the $[0.0, 0.2)$ bin containing the worst images and the $[0.8, 1.0]$ bin containing the best images. Normalizing the resulting histograms then yielded a distribution over these selection frequency bins for each class.

Next, we sampled ten images for each class from our candidate pool, following the distribution given by the class-specific selection frequency histograms. More precisely, we first computed the target number of images for each histogram bin, and then sampled from the candidates images falling into this histogram bin uniformly at random. Since we only had a limited number of images for each class, this process ran out of images for a small number of classes. In these cases, we then sampled candidate images from the next higher bin until we found a histogram bin that still had images remaining. While this slightly changes the distribution, we remark that it makes our new test set easier and only affected 0.8% of the images in the new test set.

At the end of this sampling process, we had a test set with $10,000$ images and an average sampling frequency of 0.73. This is close to the average sampling frequency of the annotated validation images (0.71).

Final Reviewing. While the methodology outlined so far closely matches the original ImageNet distribution, it is still hard to ensure that no unintended biases crept into the dataset (e.g., our MTurk workers could interpret some of the class definitions differently and select different images). So for quality control, we added a final reviewing step to our dataset creation pipeline. Its purpose was to rule out obvious biases and ensure that the dataset satisfies our quality expectations *before* we ran any models on the new dataset. This minimizes dependencies between the new test set and the existing models.

In the final reviewing step, we manually reviewed every image in the dataset. Section 3.6.1.3 includes a screenshot and brief description of the user interface. When we found an incorrect image or a near-duplicate, we removed it from the dataset. After a pass through the dataset, we then re-sampled new images from our candidate pool. In some cases, this also required new targeted Flickr searches for certain classes. We repeated this process until the dataset converged after 33 iterations. We remark that the majority of iterations only changed a small number of images.

One potential downside of the final reviewing step is that it may lead to a distribution shift. However, we accepted this possibility since we view dataset correctness to be more important than minimizing distribution shift. In the end, a test set is only interesting if it has correct labels. Note also that removing incorrect images from the dataset makes it easier, which goes *against* the main trend we observe (a drop in accuracy). Finally, we kept track of all intermediate iterations of our dataset so that we could measure the impact of this final reviewing step (see Section 3.6.3.2). This analysis shows that the main trends (a significant accuracy drop and an approximately linear relationship between original and new accuracy) also hold for the first iteration of the dataset without any additional reviewing.

3.6.2 Model performance results

After assembling our new test sets, we evaluated a broad range of models on both the original validation set and our new test sets. Section 3.11.0.1 contains a list of all models we evaluated with corresponding references and links to source code repositories. Tables 3.14 and 3.15 show the top-1 and top-5 accuracies for our main test set MatchedFrequency. Figure 3.12 visualizes the top-1 and top-5 accuracies on all three test sets.

In the main text of the chapter and Figure 3.12, we have chosen to exclude the Fisher Vector models and show accuracies only for the convolutional neural networks (convnets). Since the Fisher Vector models achieve significantly lower accuracy, a plot involving both model families would have sacrificed resolution among the convnets. We decided to focus on convnets in the main text because they have become the most widely used model family on ImageNet.

Moreover, a linear model of accuracies (as shown in previous plots) is often not a good fit when the accuracies span a wide range. Instead, a non-linear model such as a logistic or probit model can sometimes describe the data better. Indeed, this is also the case for our data on ImageNet. Figure 3.8 shows the accuracies both on a linear scale as in the previous plots, and on a *probit* scale, i.e., after applying the inverse of the Gaussian CDF

to all accuracy scores. As can be seen by comparing the two plots in Figure 3.8, the probit model is a better fit for our data. It accurately summarizes the relationship between original and new test set accuracy for all models from both model families in our testbed.

Similar to Figure 3.12, we also show the top-1 and top-5 accuracies for all three datasets in the probit domain in Figure 3.13. Section 3.7.3 describes a possible generative model that leads to a linear fit in the probit domain as exhibited by the plots in Figures 3.8 and 3.13.

3.6.3 Follow-up hypotheses

As for CIFAR-10, the gap between original and new accuracy is concerningly large. Hence we investigated multiple hypotheses for explaining this gap.

3.6.3.1 Cross validation

A natural question is whether cross-validation with the existing ImageNet data could have pointed towards a significant drop in accuracy. If adaptive overfitting to the images in the validation set is a cause for the accuracy drop, testing on different images from another cross-validation fold could produce lower accuracies.[11] Moreover, recall that the ImageNet validation set is not a strictly i.i.d. sample from the same distribution as the training set (see the beginning of Section 3.4). This also raises the question of how well a model would perform on a cross-validation fold from the training data.

To investigate these two effects, we conducted a cross-validation experiment with the ImageNet training and validation sets. In order to ensure that the new cross-validation folds contain only training images, we treated the existing validation set as one fold and created five additional folds with 50,000 images each. To this end, we drew a class-balanced sample of 250,000 images from the training set and then randomly partitioned this sample into five cross-validation folds (again class-balanced). For each of these five folds, we added the validation set (and the other training folds) to the training data so that the size of the training set was unchanged. We then trained one `resnet50` model[12] [32] for each of the five training sets and evaluated them on the corresponding held-out data. Table 3.6 shows the resulting accuracies for each split.

Overall, we do not see a large difference in accuracy on the new cross validation splits: all differences fall within the 95% confidence intervals around the accuracy scores. This is in contrast to the significantly larger accuracy drops on our new test sets.

[11] Note however that the training images may also be affected by adaptive overfitting since the model hyperparameters are often tuned for fast training speed and high training accuracy.

[12] To save computational resources, we used the optimized training code from `https://github.com/fastai/imagenet-fast`. Hence the top-5 accuracy on the original validation set is 0.4% lower than in Table 3.15.

Dataset	`resnet50` Top-5 Accuracy (%)
Original validation set	92.5 [92.3, 92.8]
Split 1	92.60 [92.4, 92.8]
Split 2	92.59 [92.4, 92.8]
Split 3	92.61 [92.4, 92.8]
Split 4	92.55 [92.3, 92.8]
Split 5	92.62 [92.4, 92.9]
New test set (MatchedFrequency)	84.7 [83.9, 85.4]

Table 3.6: `resnet50` accuracy on cross-validation splits created from the original ImageNet train and validation sets. The accuracy increase is likely caused by a small shift in distribution between the ImageNet training and validation sets.

3.6.3.2 Impact of dataset revisions

As mentioned in Section 3.6.1.4, our final reviewing pass may have led to a distribution shift compared to the original ImageNet validation set. In general, our reviewing criterion was to blacklist images that were

- not representative of the target class,
- cartoons, paintings, drawings, or renderings,
- significantly different in distribution from the original ImageNet validation set,
- unclear, blurry, severely occluded, overly edited, or including only a small target object.

For each class, our reviewing UI (screenshot in Section 3.6.1.3) displayed a random sample of ten original validation images directly next to the ten new candidate images currently chosen. At least to some extent, this allowed us to detect and correct distribution shifts between the original validation set and our candidate pool. As a concrete example, we noted in one revision of our dataset that approximately half of the images for "great white shark" were not live sharks in the water but models in museums or statues outside. In contrast, the ImageNet validation set had fewer examples of such artificial sharks. Hence we decided to remove some non-live sharks from our candidate pool and sampled new shark images as a replacement in the dataset.

Unfortunately, some of these reviewing choices are subjective. However, such choices are often an inherent part of creating a dataset and it is unclear whether a more "hands-off" approach would lead to more meaningful conclusions. For instance, if the drop in accuracy was mainly caused by a distribution shift that is easy to identify and correct (e.g., an increase in black-and-white images), the resulting drop may not be an interesting phenomenon (beyond counting black-and-white images). Hence we decided to *both* remove

distribution shifts that we found easy to identify visually, and also to measure the effect of these interventions.

Our reviewing process was iterative, i.e., we made a full pass over every incomplete class in a given dataset revision before sampling new images to fill the resulting gaps. Each time we re-sampled our dataset, we saved the current list of images in our version control system. This allowed us to track the datasets over time and later measure the model accuracy for each dataset revision. We remark that we only computed model accuracies on intermediate revisions after we had arrived at the final revision of the corresponding dataset.

Figure 3.9 plots the top-1 accuracy of a `resnet50` model versus the dataset revision for our new MatchedFrequency test set. Overall, reviewing improved model accuracy by about 4% for this dataset. This is evidence that our manual reviewing did not cause the drop in accuracy between the original and new dataset.

In addition, we also investigated whether the linear relationship between original and new test accuracy was affected by our final reviewing passes. To this end, we evaluated our model testbed on the first revision of our MatchedFrequency test set. As can be seen in Figure 3.10, the resulting accuracies still show a good linear fit that is of similar quality as in Figure 3.12. This shows that the linear relationship between the test accuracies is not a result of our reviewing intervention.

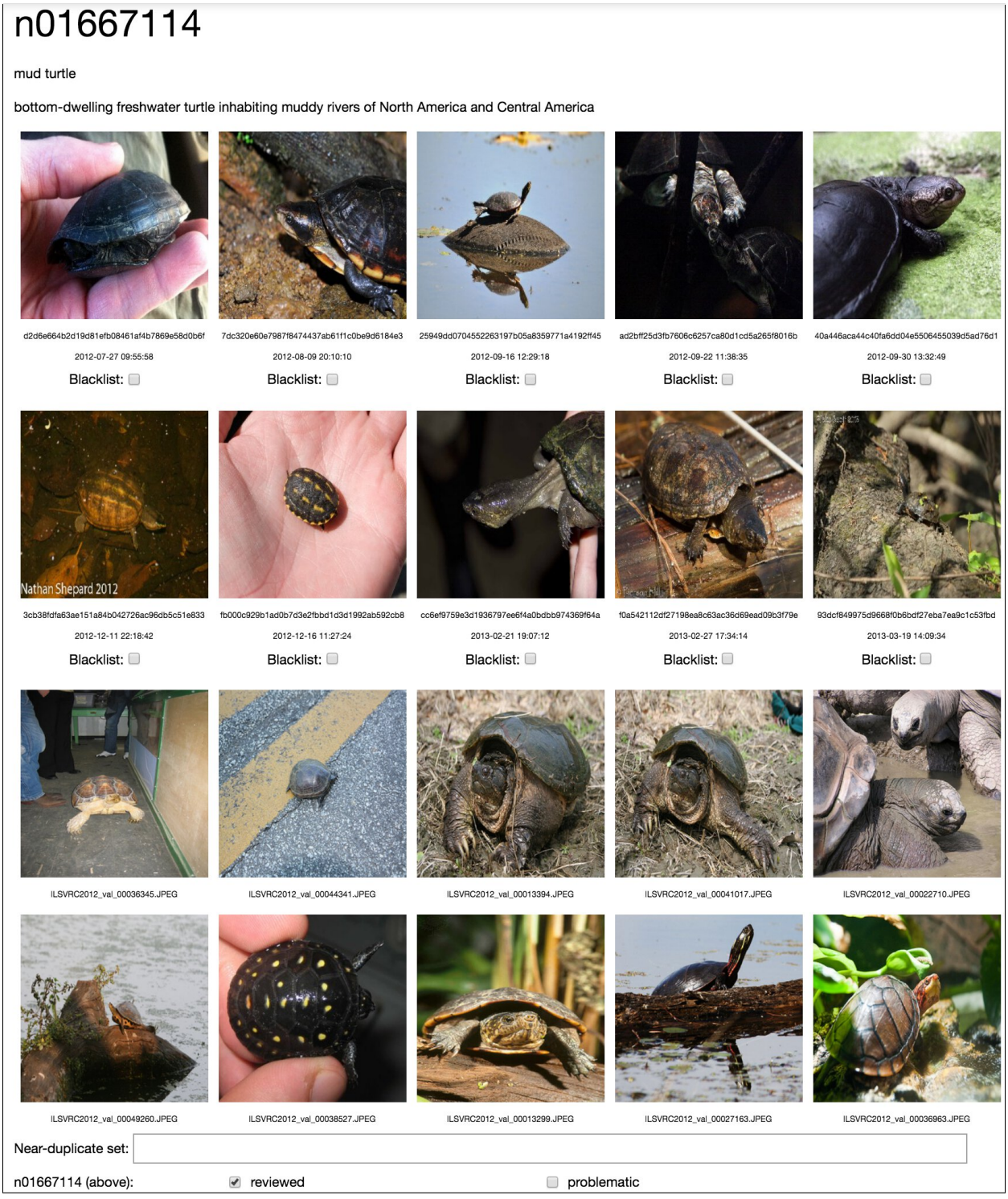


Figure 3.7: The user interface we built to review dataset revisions and remove incorrect or near duplicate images. This user interface was not used for MTurk but only in the final dataset review step conducted by the authors of this paper.

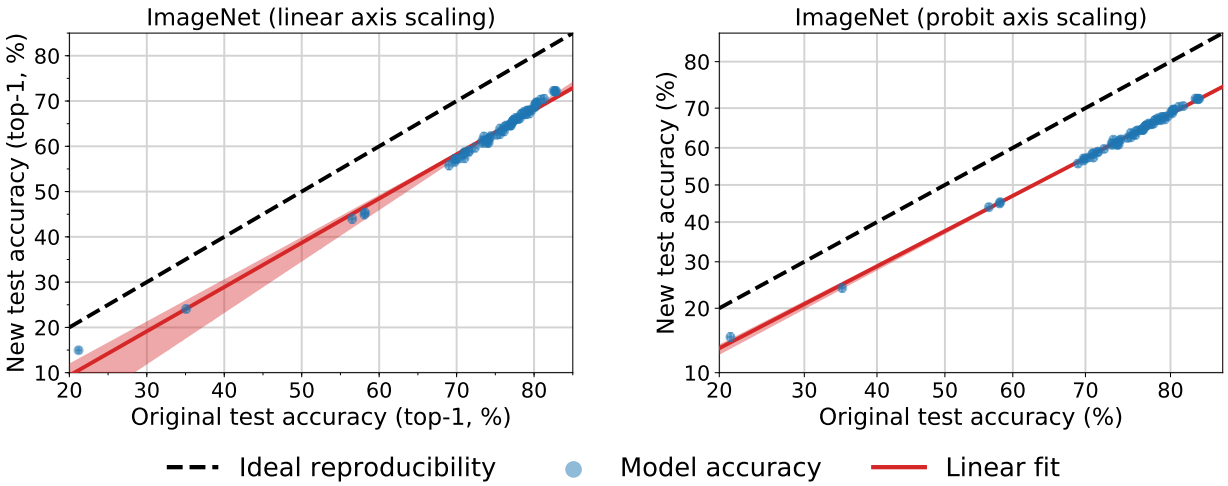


Figure 3.8: Model accuracy on the original ImageNet validation set vs. our new test set MatchedFrequency. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals), and we now also include the Fisher Vector models. The left plot shows the model accuracies with a linear scale on the axes. The right plot instead uses a *probit* scale, i.e., accuracy α appears at $\Phi^{-1}(\alpha)$, where Φ is the Gaussian CDF. Comparing the two plot provides evidence that the probit model is a better fit for the accuracy scores. Over a range of 60 percentage points, the linear fit in the probit domain accurately describes the relationship between original and new test set accuracy. The shaded region around the linear fit is a 95% confidence region from 100,000 bootstrap samples. The confidence region is present in both plots but is significantly smaller in the right plot due to the better fit in the probit domain.

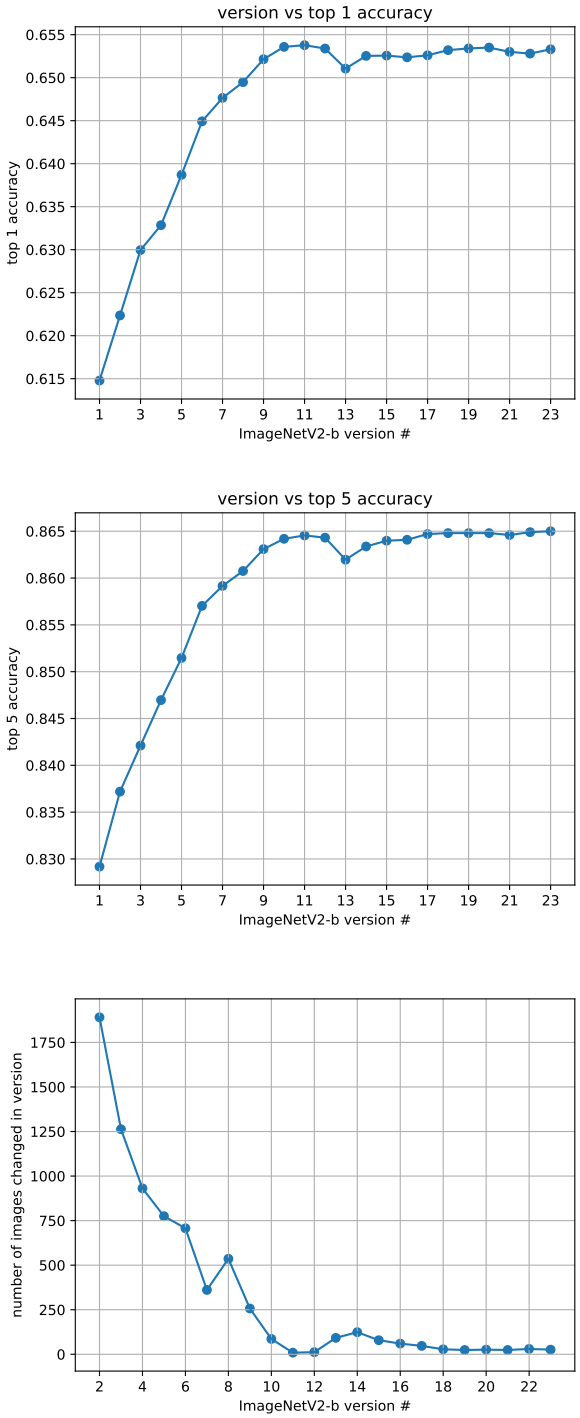


Figure 3.9: Impact of the reviewing passes on the accuracy of a `resnet152` on our new MatchedFrequency test set. The revision numbers correspond to the chronological ordering in which we created the dataset revisions

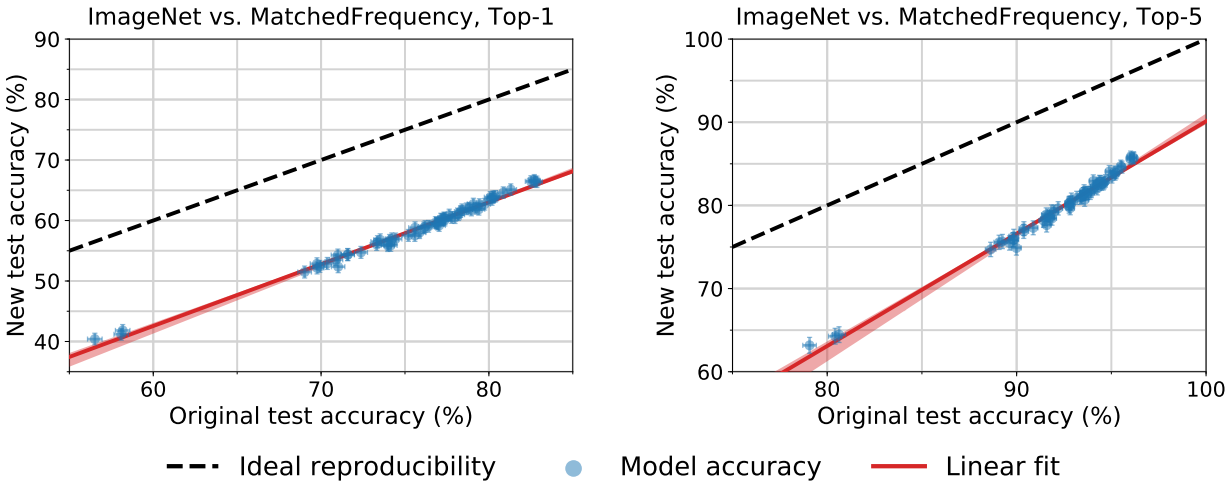


Figure 3.10: Model accuracy on the original ImageNet validation set vs. accuracy on *the first revision* of our MatchedFrequency test set. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples. The plots show that the linear relationship between original and new test accuracy also occurs without our final dataset reviewing step. The accuracy plots for the final revision of MatchedFrequency can be found in Figure 3.12.

3.7 Discussion

We now return to the main question from Section 3.2: *What causes the accuracy drops?* As before, we distinguish between two possible mechanisms.

3.7.1 Adaptivity gap

In its prototypical form, *adaptive* overfitting would manifest itself in diminishing returns observed on the new test set (see Section 3.2.1). However, we do not observe this pattern on either CIFAR-10 or ImageNet. On both datasets, the slope of the linear fit is *greater* than 1, i.e., each point of accuracy improvement on the original test set translates to more than 1% on the new test set. This is the opposite of the standard overfitting scenario. So at least on CIFAR-10 and ImageNet, multiple years of competitive test set adaptivity did not lead to diminishing accuracy numbers.

While our experiments rule out the most dangerous form of adaptive overfitting, we remark that they do not exclude all variants. For instance, it could be that any test set adaptivity leads to a roughly constant drop in accuracy. Then all models are affected equally and we would see no diminishing returns since later models could still be better. Testing for this form of adaptive overfitting likely requires a new test set that is truly i.i.d. and not the result of a separate data collection effort. Finding a suitable dataset for such an experiment is an interesting direction for future research.

The lack of adaptive overfitting contradicts conventional wisdom in machine learning. We now describe two mechanisms that could have prevented adaptive overfitting:

The Ladder mechanism. Blum and Hardt introduced the Ladder algorithm to protect machine learning competitions against adaptive overfitting [3]. The core idea is that constrained interaction with the test set can allow a large number of model evaluations to succeed, even if the models are chosen adaptively. Due to the natural form of their algorithm, the authors point out that it can also be seen as a mechanism that the machine learning community *implicitly* follows.

Limited model class. Adaptivity is only a problem if we can choose among models for which the test set accuracy differs significantly from the population accuracy. Importantly, this argument does not rely on the number of *all* possible models (e.g., all parameter settings of a neural network), but only on those models that could actually be evaluated on the test set. For instance, the standard deep learning workflow only produces models trained with SGD-style algorithms on a fixed training set, and requires that the models achieve high training accuracy (otherwise we would not consider the corresponding hyperparameters). Hence the number of different models arising from the current methodology may be small enough so that uniform convergence holds.

Our experiments offer little evidence for favoring one explanation over the other. One observation is that the convolutional networks shared many errors on CIFAR-10, which could

be an indicator that the models are rather similar. But to gain a deeper understanding into adaptive overfitting, it is likely necessary to gather further data from more machine learning benchmarks, especially in scenarios where adaptive overfitting *does* occur naturally.

3.7.2 Distribution gap

The lack of diminishing returns in our experiments points towards the distribution gap as the primary reason for the accuracy drops. Moreover, our results on ImageNet show that changes in the sampling strategy can indeed affect model accuracies by a large amount, even if the data source and other parts of the dataset creation process stay the same.

So in spite of our efforts to match the original dataset creation process, the distribution gap is still our leading hypothesis for the accuracy drops. This demonstrates that it is surprisingly hard to accurately replicate the distribution of current image classification datasets. The main difficulty likely is the subjective nature of the human annotation step. There are many parameters that can affect the quality of human labels such as the annotator population (MTurk vs. students, qualifications, location & time, etc.), the exact task format, and compensation. Moreover, there are no exact definitions for many classes in ImageNet (e.g., see Section 3.11.0.6). Understanding these aspects in more detail is an important direction for designing future datasets that contain challenging images while still being labeled correctly.

The difficulty of clearly defining the data distribution, combined with the brittle behavior of the tested models, calls into question whether the black-box and i.i.d. framework of learning can produce reliable classifiers. Our analysis of selection frequencies in Figure 3.15 (Section 3.11.0.5) shows that we could create a new test set with even lower model accuracies. The images in this hypothetical dataset would still be correct, from Flickr, and selected by more than half of the MTurk labelers on average. So in spite of the impressive accuracy scores on the original validation set, current ImageNet models still have difficulty generalizing from "easy" to "hard" images.

3.7.3 A model for the linear fit

Finally, we briefly comment on the striking linear relationship between original and new test accuracies that we observe in all our experiments (for instance, see Figure 3.1 in the introduction or Figures 3.12 and 3.13 in the supplementary material at the end of the chapter). To illustrate how this phenomenon could arise, we present a simple data model where a small modification of the data distribution can lead to significant changes in accuracy, yet the relative order of models is preserved as a linear relationship. We emphasize that this model should not be seen as the true explanation. Instead, we hope it can inform future experiments that explore natural variations in test distributions.

First, as we describe in Section 3.6.2, we find that we achieve better fits to our data under a *probit scaling* of the accuracies. Over a wide range from 21% to 83% (all models in our ImageNet testbed), the accuracies on the new test set, α_{new}, are related to the accuracies on

the original test set, α_{orig}, by the relationship

$$\Phi^{-1}(\alpha_{\text{new}}) \;=\; u \cdot \Phi^{-1}(\alpha_{\text{orig}}) + v$$

where Φ is the Gaussian CDF, and u and v are scalars. The probit scale is in a sense more natural than a linear scale as the accuracy numbers are probabilities. When we plot accuracies on a probit scale in Figures 3.8 and 3.13, we effectively visualize $\Phi^{-1}(\alpha)$ instead of α.

We now provide a simple plausible model where the original and new accuracies are related linearly on a probit scale. Assume that every example i has a scalar "difficulty" $\tau_i \in \mathbb{R}$ that quantifies how easy it is to classify. Further assume the probability of a model j correctly classifying an image with difficulty τ is given by an increasing function $\zeta_j(\tau)$. We show that for restricted classes of difficulty functions ζ_j, we find a linear relationship between average accuracies after distribution shifts.

To be specific, we focus on the following parameterization. Assume the difficulty distribution of images in a test set follows a normal distribution with mean μ and variance σ^2. Further assume that

$$\zeta_j(\tau) \;=\; \Phi(s_j - \tau)\;,$$

where $\Phi : \mathbb{R} \to (0, 1)$ is the CDF of a standard normal distribution, and s_j is the "skill" of model j. Models with higher skill have higher classification accuracy, and images with higher difficulty lead to smaller classification accuracy. Again, the choice of Φ here is somewhat arbitrary: any sigmoidal function that maps $(-\infty, +\infty)$ to $(0, 1)$ is plausible. But using the Gaussian CDF yields a simple calculation illustrating the linear phenomenon.

Using the above notation, the accuracy $\alpha_{j,\mu,\sigma}$ of a model j on a test set with difficulty mean μ and variance σ is then given by

$$\alpha_{j,\mu,\sigma} \;=\; \mathop{\mathbb{E}}_{\tau \sim \mathcal{N}(\mu,\sigma)} \left[\Phi(s_j - \tau)\right]\;.$$

We can expand the CDF into an expectation and combine the two expectations by utilizing the fact that a linear combination of two Gaussians is again Gaussian. This yields:

$$\alpha_{j,\mu,\sigma} \;=\; \Phi\left(\frac{s_j - \mu}{\sqrt{\sigma^2 + 1}}\right)\;.$$

On a probit scale, the quantities we plot are given by

$$\tilde{\alpha}_{j,\mu,\sigma} \;=\; \Phi^{-1}(\alpha_{j,\mu,\sigma}) \;=\; \frac{s_j - \mu}{\sqrt{\sigma^2 + 1}}\;.$$

Next, we consider the case where we have multiple models and two test sets with difficulty parameters μ_k and σ_k respectively for $k \in \{1, 2\}$. Then $\tilde{\alpha}_{j,2}$, the probit-scaled accuracy on the second test set, is a linear function of the accuracy on the first test set, $\tilde{\alpha}_{j,1}$:

$$\tilde{\alpha}_{j,2} \;=\; u \cdot \tilde{\alpha}_{j,1} + v\;,$$

with

$$u = \frac{\sqrt{\sigma_1^2 + 1}}{\sqrt{\sigma_2^2 + 1}} \quad \text{and} \quad v = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_2^2 + 1}} \,.$$

Hence, we see that the Gaussian difficulty model above yields a linear relationship between original and new test accuracy in the probit domain. While the Gaussian assumptions here made the calculations simple, a variety of different simple classes of ζ_j will give rise to the same linear relationship between the accuracies on two different test sets.

3.8 Related work

We now briefly discuss related threads in machine learning. To the best of our knowledge, there are no reproducibility experiments directly comparable to ours in the literature.

Dataset biases. The computer vision community has a rich history of creating new datasets and discussing their relative merits, e.g., [25, 99, 72, 92, 23, 15, 81, 57]. The paper closest to ours is [92], which studies dataset biases by measuring how models trained on one dataset generalize to other datasets. The main difference to our work is that the authors test generalization across *different* datasets, where larger changes in the distribution (and hence larger drops in accuracy) are expected. In contrast, our experiments explicitly attempt to reproduce the original data distribution and demonstrate that even small variations arising in this process can lead to significant accuracy drops. Moreover, [92] do not test on previously unseen data, so their experiments cannot rule out adaptive overfitting.

Transfer learning from ImageNet. Kornblith et. al. [51] study how well accuracy on ImageNet transfers to other image classification datasets. An important difference from both our work and [92] is that the the ImageNet models are re-trained on the target datasets. The authors find that better ImageNet models usually perform better on the target dataset as well. Similar to [92], these experiments cannot rule out adaptive overfitting since the authors do not use new data. Moreover, the experiments do not measure accuracy drops due to small variations in the data generating process since the models are evaluated on a different task with an explicit adaptation step. Interestingly, the authors also find an approximately linear relationship between ImageNet and transfer accuracy.

Adversarial examples. While adversarial examples [88, 2] also show that existing models are brittle, the perturbations have to be finely tuned since models are much more robust to random perturbations. In contrast, our results demonstrate that even small, benign variations in the data sampling process can already lead to a significant accuracy drop without an adversary.

A natural question is whether adversarially robust models are also more robust to the distribution shifts observed in our work. As a first data point, we tested the common ℓ_∞-robustness baseline from [61] for CIFAR-10. Interestingly, the accuracy numbers of this model fall almost exactly on the linear fit given by the other models in our testbed. Hence ℓ_∞-robustness does not seem to offer benefits for the distribution shift arising from our reproducibility experiment. However, we note that more forms of adversarial robustness such as spatial transformations or color space changes have been studied [21, 37, 95, 24, 45]. Testing these variants is an interesting direction for future work.

Non-adversarial image perturbations. Recent work also explores less adversarial changes to the input, e.g., [29, 35]. In these papers, the authors modify the ImageNet validation set via well-specified perturbations such as Gaussian noise, a fixed rotation, or adding a synthetic snow-like pattern. Standard ImageNet models then achieve significantly lower accuracy on the perturbed examples than on the unmodified validation set. While this is an interesting test of robustness, the mechanism underlying the accuracy drops is significantly different from our work. The aforementioned papers rely on an intentional, clearly-visible, and well-defined perturbation of existing validation images. Moreover, some of the interventions are quite different from the ImageNet validation set (e.g., ImageNet contains few images of falling snow). In contrast, our experiments use new images and match the distribution of the existing validation set as closely as possible. Hence it is unclear what properties of our new images cause the accuracy drops.

3.9 Conclusion and future work

The expansive growth of machine learning rests on the aspiration to deploy trained systems in a variety of challenging environments. Common examples include autonomous vehicles, content moderation, and medicine. In order to use machine learning in these areas responsibly, it is important that we can both train models with sufficient generalization abilities, and also reliably measure their performance. As our results show, these goals still pose significant hurdles even in a benign environment.

Our experiments are only a first step in addressing this reliability challenge. One important question is whether other machine learning tasks are also resilient to adaptive overfitting, but similarly brittle under natural variations in the data. Another direction is developing methods for more comprehensive yet still realistic evaluations of machine learning systems. Of course, the overarching goal is to develop learning algorithms that generalize reliably. While this is often a vague goal, our new test sets offer a well-defined instantiation of this challenge that is beyond the reach of current methods. Generalizing from our "easy" to slightly "harder" images will hopefully serve as a starting point towards a future generation of more reliable models

3.10 Supplementary material for CIFAR-10

In this section we provide supplementary material related to our CIFAR-10 experiments.

3.10.0.1 Keyword Distribution in CIFAR-10

The sub-tables in Table 3.7 show the keyword distribution for each of the ten classes in the original CIFAR-10 test set and our new test set.

Table 3.7: Distribution of the top 25 keywords in each class for the new and original test set.

Frog		
	New	Original
bufo_bufo	0.64%	0.63%
leopard_frog	0.64%	0.64%
bufo_viridis	0.59%	0.57%
rana_temporaria	0.54%	0.53%
bufo	0.49%	0.47%
bufo_americanus	0.49%	0.46%
toad	0.49%	0.46%
green_frog	0.45%	0.44%
rana_catesbeiana	0.45%	0.43%
bufo_marinus	0.45%	0.43%
bullfrog	0.45%	0.42%
american_toad	0.45%	0.43%
frog	0.35%	0.35%
rana_pipiens	0.35%	0.32%
toad_frog	0.30%	0.30%
spadefoot	0.30%	0.27%
western_toad	0.30%	0.26%
grass_frog	0.30%	0.27%
pickerel_frog	0.25%	0.24%
spring_frog	0.25%	0.22%
rana_clamitans	0.20%	0.20%
natterjack	0.20%	0.17%
crapaud	0.20%	0.18%
bufo_calamita	0.20%	0.18%
alytes_obstetricans	0.20%	0.16%

Cat		
	New	Original
tabby_cat	1.78%	1.78%
tabby	1.53%	1.52%
domestic_cat	1.34%	1.33%
cat	1.24%	1.25%
house_cat	0.79%	0.79%
felis_catus	0.69%	0.69%
mouser	0.64%	0.63%
felis_domesticus	0.54%	0.50%
true_cat	0.49%	0.47%
tomcat	0.49%	0.49%
alley_cat	0.30%	0.30%
felis_bengalensis	0.15%	0.11%
nougat	0.10%	0.05%
gray	0.05%	0.03%
manx_cat	0.05%	0.04%
fissiped	0.05%	0.03%
persian_cat	0.05%	0.03%
puss	0.05%	0.05%
catnap	0.05%	0.03%
tiger_cat	0.05%	0.03%
black_cat	0.05%	0.04%
bedspread	0.00%	0.02%
siamese_cat	0.00%	0.02%
tortoiseshell	0.00%	0.02%
kitty-cat	0.00%	0.02%

Dog		
	New	Original
pekingese	1.24%	1.22%
maltese	0.94%	0.93%
puppy	0.89%	0.87%
chihuahua	0.84%	0.81%
dog	0.69%	0.67%
pekinese	0.69%	0.66%
toy_spaniel	0.59%	0.60%
mutt	0.49%	0.47%
mongrel	0.49%	0.49%
maltese_dog	0.45%	0.43%
toy_dog	0.40%	0.36%
japanese_spaniel	0.40%	0.38%
blenheim_spaniel	0.35%	0.35%
english_toy_spaniel	0.35%	0.31%
domestic_dog	0.35%	0.32%
peke	0.30%	0.28%
canis_familiaris	0.30%	0.27%
lapdog	0.30%	0.30%
king_charles_spaniel	0.20%	0.17%
toy	0.15%	0.13%
feist	0.10%	0.06%
pet	0.10%	0.07%
cavalier	0.10%	0.05%
canine	0.05%	0.04%
cur	0.05%	0.04%

Deer		
	New	Original
elk	0.79%	0.77%
capreolus_capreolus	0.74%	0.71%
cervus_elaphus	0.64%	0.61%
fallow_deer	0.64%	0.63%
roe_deer	0.59%	0.60%
deer	0.59%	0.60%
muntjac	0.54%	0.51%
mule_deer	0.54%	0.51%
odocoileus_hemionus	0.49%	0.50%
fawn	0.49%	0.49%
alces_alces	0.40%	0.36%
wapiti	0.40%	0.36%
american_elk	0.40%	0.35%
red_deer	0.35%	0.33%
moose	0.35%	0.35%
rangifer_caribou	0.25%	0.24%
rangifer_tarandus	0.25%	0.24%
caribou	0.25%	0.23%
sika	0.25%	0.22%
woodland_caribou	0.25%	0.21%
dama_dama	0.20%	0.19%
cervus_sika	0.20%	0.16%
barking_deer	0.20%	0.18%
sambar	0.15%	0.15%
stag	0.15%	0.13%

Bird		
	New	Original
cassowary	0.89%	0.85%
bird	0.84%	0.84%
wagtail	0.74%	0.74%
ostrich	0.69%	0.68%
struthio_camelus	0.54%	0.51%
sparrow	0.54%	0.52%
emu	0.54%	0.51%
pipit	0.49%	0.47%
passerine	0.49%	0.50%
accentor	0.49%	0.49%
honey_eater	0.40%	0.37%
dunnock	0.40%	0.37%
alauda_arvensis	0.30%	0.26%
nandu	0.30%	0.27%
prunella_modularis	0.30%	0.30%
anthus_pratensis	0.30%	0.28%
finch	0.25%	0.24%
lark	0.25%	0.20%
meadow_pipit	0.25%	0.20%
rhea_americana	0.25%	0.21%
flightless_bird	0.15%	0.10%
emu_novaehollandiae	0.15%	0.12%
dromaius_novaehollandiae	0.15%	0.14%
apteryx	0.15%	0.10%
flying_bird	0.15%	0.13%

Ship		
	New	Original
passenger_ship	0.79%	0.78%
boat	0.64%	0.64%
cargo_ship	0.40%	0.37%
cargo_vessel	0.40%	0.39%
pontoon	0.35%	0.31%
container_ship	0.35%	0.31%
speedboat	0.35%	0.32%
freighter	0.35%	0.32%
pilot_boat	0.35%	0.31%
ship	0.35%	0.31%
cabin_cruiser	0.30%	0.29%
police_boat	0.30%	0.25%
sea_boat	0.30%	0.29%
oil_tanker	0.30%	0.29%
pleasure_boat	0.25%	0.21%
lightship	0.25%	0.22%
powerboat	0.25%	0.25%
guard_boat	0.25%	0.20%
dredger	0.25%	0.20%
hospital_ship	0.25%	0.21%
banana_boat	0.20%	0.19%
merchant_ship	0.20%	0.17%
liberty_ship	0.20%	0.15%
container_vessel	0.20%	0.19%
tanker	0.20%	0.18%

Truck		
	New	Original
dump_truck	0.89%	0.89%
trucking_rig	0.79%	0.76%
delivery_truck	0.64%	0.61%
truck	0.64%	0.65%
tipper_truck	0.64%	0.60%
camion	0.59%	0.58%
fire_truck	0.59%	0.55%
lorry	0.54%	0.53%
garbage_truck	0.54%	0.53%
moving_van	0.35%	0.32%
tractor_trailer	0.35%	0.34%
tipper	0.35%	0.30%
aerial_ladder_truck	0.35%	0.34%
ladder_truck	0.30%	0.26%
fire_engine	0.30%	0.27%
dumper	0.30%	0.28%
trailer_truck	0.30%	0.28%
wrecker	0.30%	0.27%
articulated_lorry	0.25%	0.24%
tipper_lorry	0.25%	0.25%
semi	0.20%	0.18%
sound_truck	0.15%	0.12%
tow_truck	0.15%	0.12%
delivery_van	0.15%	0.11%
bookmobile	0.10%	0.10%

Horse		
	New	Original
arabian	1.14%	1.12%
lipizzan	1.04%	1.02%
broodmare	0.99%	0.97%
gelding	0.74%	0.73%
quarter_horse	0.74%	0.72%
stud_mare	0.69%	0.69%
lippizaner	0.54%	0.52%
appaloosa	0.49%	0.45%
lippizan	0.49%	0.46%
dawn_horse	0.45%	0.42%
stallion	0.45%	0.43%
tennessee_walker	0.45%	0.45%
tennessee_walking_horse	0.40%	0.38%
walking_horse	0.30%	0.28%
riding_horse	0.20%	0.20%
saddle_horse	0.20%	0.18%
female_horse	0.15%	0.11%
cow_pony	0.15%	0.11%
male_horse	0.15%	0.14%
buckskin	0.15%	0.13%
horse	0.10%	0.08%
equine	0.10%	0.08%
quarter	0.10%	0.07%
cavalry_horse	0.10%	0.09%
thoroughbred	0.10%	0.06%

Airplane		
	New	Original
stealth_bomber	0.94%	0.92%
airbus	0.89%	0.89%
stealth_fighter	0.79%	0.80%
fighter_aircraft	0.79%	0.76%
biplane	0.74%	0.74%
attack_aircraft	0.69%	0.67%
airliner	0.64%	0.61%
jetliner	0.59%	0.56%
monoplane	0.54%	0.55%
twinjet	0.54%	0.52%
dive_bomber	0.54%	0.52%
jumbo_jet	0.49%	0.47%
jumbojet	0.35%	0.35%
propeller_plane	0.30%	0.28%
fighter	0.20%	0.20%
plane	0.20%	0.15%
amphibious_aircraft	0.20%	0.20%
multiengine_airplane	0.15%	0.14%
seaplane	0.15%	0.14%
floatplane	0.10%	0.05%
multiengine_plane	0.10%	0.06%
reconnaissance_plane	0.10%	0.09%
airplane	0.10%	0.08%
tail	0.10%	0.05%
joint	0.05%	0.04%

Automobile		
	New	Original
coupe	1.29%	1.26%
convertible	1.19%	1.18%
station_wagon	0.99%	0.98%
automobile	0.89%	0.90%
car	0.84%	0.81%
auto	0.84%	0.83%
compact_car	0.79%	0.76%
shooting_brake	0.64%	0.63%
estate_car	0.59%	0.59%
wagon	0.54%	0.51%
police_cruiser	0.45%	0.45%
motorcar	0.40%	0.40%
taxi	0.20%	0.17%
cruiser	0.15%	0.13%
compact	0.15%	0.11%
beach_wagon	0.15%	0.13%
funny_wagon	0.10%	0.05%
gallery	0.10%	0.07%
cab	0.10%	0.07%
ambulance	0.10%	0.07%
door	0.00%	0.03%
ford	0.00%	0.03%
opel	0.00%	0.03%
sport_car	0.00%	0.03%
sports_car	0.00%	0.03%

3.10.0.2 Full List of Models Evaluated on CIFAR-10

The following list contains all models we evaluated on CIFAR-10 with references and links to the corresponding source code.

1. autoaug_pyramid_net [13, 31] https://github.com/tensorflow/models/tree/master/research/autoaugment
2. autoaug_shake_shake_112 [13, 28] https://github.com/tensorflow/models/tree/master/research/autoaugment
3. autoaug_shake_shake_32 [13, 28] https://github.com/tensorflow/models/tree/master/research/autoaugment
4. autoaug_shake_shake_96 [13, 28] https://github.com/tensorflow/models/tree/master/research/autoaugment
5. autoaug_wrn [13, 102] https://github.com/tensorflow/models/tree/master/research/autoaugment
6. cudaconvnet [54] https://github.com/akrizhevsky/cuda-convnet2
7. darc [48] http://lis.csail.mit.edu/code/gdl.html
8. densenet_BC_100_12 [40] https://github.com/hysts/pytorch_image_classification/
9. nas [104] https://github.com/tensorflow/models/blob/master/research/slim/nets/nasnet/nasnet.py#L32
10. pyramidnet_basic_110_270 [31] https://github.com/hysts/pytorch_image_classification/
11. pyramidnet_basic_110_84 [31] https://github.com/hysts/pytorch_image_classification/
12. random_features_256k_aug [10] https://github.com/modestyachts/nondeep Random 1 layer convolutional network with 256k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6, and horizontal flip data augmentation.
13. random_features_256k [10] https://github.com/modestyachts/nondeep Random 1 layer convolutional network with 256k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6.
14. random_features_32k_aug [10] https://github.com/modestyachts/nondeep Random 1 layer convolutional network with 32k filters sampled from image patches, patch size = 6, pool size 15, pool stride 6, and horizontal flip data augmentation.

15. random_features_32k [10] Random 1 layer convolutional network with 32k filters sampled from image patches, patch size = 6, pool size 15, pool stride 16.
16. resnet_basic_32 [32] https://github.com/hysts/pytorch_image_classification/
17. resnet_basic_44 [32] https://github.com/hysts/pytorch_image_classification/
18. resnet_basic_56 [32] https://github.com/hysts/pytorch_image_classification/
19. resnet_basic_110 [32] https://github.com/hysts/pytorch_image_classification/
20. resnet_preact_basic_110 [34] https://github.com/hysts/pytorch_image_classification/
21. resnet_preact_bottleneck_164 [34] https://github.com/hysts/pytorch_image_classification/
22. resnet_preact_tf [34] https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet
23. resnext_29_4x64d [96] https://github.com/hysts/pytorch_image_classification/
24. resnext_29_8x64d [96] https://github.com/hysts/pytorch_image_classification/
25. shake_drop [98] https://github.com/imenurok/ShakeDrop
26. shake_shake_32d [28] https://github.com/hysts/pytorch_image_classification/
27. shake_shake_64d [28] https://github.com/hysts/pytorch_image_classification/
28. shake_shake_96d [28] https://github.com/hysts/pytorch_image_classification/
29. shake_shake_64d_cutout [28, 16] https://github.com/hysts/pytorch_image_classification/
30. vgg16_keras [84, 59] https://github.com/geifmany/cifar-vgg
31. vgg_15_BN_64 [84, 59] https://github.com/hysts/pytorch_image_classification/

32. `wide_resnet_tf` [102] `https://github.com/tensorflow/models/tree/b871670b5ae29aaa6cad1b2d4e004882f716c466/resnet`

33. `wide_resnet_28_10` [102] `https://github.com/hysts/pytorch_image_classification/`

34. `wide_resnet_28_10_cutout` [102, 16] `https://github.com/hysts/pytorch_image_classification/`

3.10.0.3 Full Results Table

Table 3.12 contains the detailed accuracy scores for the original CIFAR-10 test set and our new test set.

3.10.0.4 Full Results Table for the Exactly Class-Balanced Test Set

Table 3.13 contains the detailed accuracy scores for the original CIFAR-10 test set and the exactly class-balanced variant of our new test set.

3.10.0.5 Hard Images

Figure 3.11 shows the images in our new CIFAR-10 test set that were misclassified by all models in our testbed. As can be seen in the figure, the class labels for these images are correct.

Figure 3.11: Hard images from our new test set that no model correctly. The caption of each image states the correct class label ("True") and the label predicted by most models ("Predicted").

Table 3.12: Model accuracy on the original CIFAR-10 test set and our new test set. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.10.0.2.

CIFAR-10						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_tf	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	autoaug_shake_shake_112	98.1 [97.8, 98.4]	93.9 [92.7, 94.9]	4.3	2	0
3	autoaug_shake_shake_96_	98.0 [97.7, 98.3]	93.7 [92.6, 94.7]	4.3	3	0
4	autoaug_wrn_tf	97.5 [97.1, 97.8]	93.0 [91.8, 94.1]	4.4	4	0
5	autoaug_shake_shake_32_	97.3 [97.0, 97.6]	92.9 [91.7, 94.0]	4.4	6	-1
6	shake_shake_64d_cutout	97.1 [96.8, 97.4]	93.0 [91.8, 94.1]	4.1	5	1
7	shake_shake_26_2x96d_SS	97.1 [96.7, 97.4]	91.9 [90.7, 93.1]	5.1	9	-2
8	shake_shake_64d	97.0 [96.6, 97.3]	91.4 [90.1, 92.6]	5.6	10	-2
9	wrn_28_10_cutout16	97.0 [96.6, 97.3]	92.0 [90.7, 93.1]	5.0	8	1
10	shake_drop	96.9 [96.5, 97.2]	92.3 [91.0, 93.4]	4.6	7	3
11	shake_shake_32d	96.6 [96.2, 96.9]	89.8 [88.4, 91.1]	6.8	13	-2
12	darc	96.6 [96.2, 96.9]	89.5 [88.1, 90.8]	7.1	16	-4
13	resnext_29_4x64d	96.4 [96.0, 96.7]	89.6 [88.2, 90.9]	6.8	15	-2
14	pyramidnet_basic_110_27	96.3 [96.0, 96.7]	90.5 [89.1, 91.7]	5.9	11	3
15	resnext_29_8x64d	96.2 [95.8, 96.6]	90.0 [88.6, 91.2]	6.3	12	3
16	wrn_28_10	95.9 [95.5, 96.3]	89.7 [88.3, 91.0]	6.2	14	2
17	pyramidnet_basic_110_84	95.7 [95.3, 96.1]	89.3 [87.8, 90.6]	6.5	17	0
18	densenet_BC_100_12	95.5 [95.1, 95.9]	87.6 [86.1, 89.0]	8.0	20	-2
19	nas	95.4 [95.0, 95.8]	88.8 [87.4, 90.2]	6.6	18	1
20	wide_resnet_tf_28_10	95.0 [94.6, 95.4]	88.5 [87.0, 89.9]	6.5	19	1
21	resnet_v2_bottleneck_16	94.2 [93.7, 94.6]	85.9 [84.3, 87.4]	8.3	22	-1
22	vgg16_keras	93.6 [93.1, 94.1]	85.3 [83.6, 86.8]	8.3	23	-1
23	resnet_basic_110	93.5 [93.0, 93.9]	85.2 [83.5, 86.7]	8.3	24	-1
24	resnet_v2_basic_110	93.4 [92.9, 93.9]	86.5 [84.9, 88.0]	6.9	21	3
25	resnet_basic_56	93.3 [92.8, 93.8]	85.0 [83.3, 86.5]	8.3	25	0
26	resnet_basic_44	93.0 [92.5, 93.5]	84.2 [82.6, 85.8]	8.8	29	-3
27	vgg_15_BN_64	93.0 [92.5, 93.5]	84.9 [83.2, 86.4]	8.1	27	0
28	resnetv2_tf_32	92.7 [92.2, 93.2]	84.4 [82.7, 85.9]	8.3	28	0
29	resnet_basic_32	92.5 [92.0, 93.0]	84.9 [83.2, 86.4]	7.7	26	3
30	cudaconvnet	88.5 [87.9, 89.2]	77.5 [75.7, 79.3]	11.0	30	0
31	random_features_256k_au	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12.5	31	0
32	random_features_32k_aug	85.0 [84.3, 85.7]	71.9 [69.9, 73.9]	13.0	32	0
33	random_features_256k	84.2 [83.5, 84.9]	69.9 [67.8, 71.9]	14.3	33	0
34	random_features_32k	83.3 [82.6, 84.0]	67.9 [65.9, 70.0]	15.4	34	0

Table 3.13: Model accuracy on the original CIFAR-10 test set and the exactly class-balanced variant of our new test set. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.10.0.2.

CIFAR-10						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	autoaug_pyramid_net_tf	98.4 [98.1, 98.6]	95.5 [94.5, 96.4]	2.9	1	0
2	autoaug_shake_shake_112	98.1 [97.8, 98.4]	94.0 [92.9, 95.0]	4.1	2	0
3	autoaug_shake_shake_96_	98.0 [97.7, 98.3]	93.9 [92.8, 94.9]	4.1	3	0
4	autoaug_wrn_tf	97.5 [97.1, 97.8]	93.0 [91.8, 94.1]	4.5	6	-2
5	autoaug_shake_shake_32_	97.3 [97.0, 97.6]	93.2 [92.0, 94.2]	4.2	4	1
6	shake_shake_64d_cutout	97.1 [96.8, 97.4]	93.1 [91.9, 94.2]	4.0	5	1
7	shake_shake_26_2x96d_SS	97.1 [96.7, 97.4]	92.0 [90.7, 93.1]	5.1	9	-2
8	shake_shake_64d	97.0 [96.6, 97.3]	91.9 [90.6, 93.1]	5.1	10	-2
9	wrn_28_10_cutout16	97.0 [96.6, 97.3]	92.1 [90.8, 93.2]	4.9	8	1
10	shake_drop	96.9 [96.5, 97.2]	92.3 [91.1, 93.4]	4.6	7	3
11	shake_shake_32d	96.6 [96.2, 96.9]	90.0 [88.6, 91.3]	6.6	15	-4
12	darc	96.6 [96.2, 96.9]	89.9 [88.5, 91.2]	6.7	16	-4
13	resnext_29_4x64d	96.4 [96.0, 96.7]	90.1 [88.8, 91.4]	6.2	12	1
14	pyramidnet_basic_110_27	96.3 [96.0, 96.7]	90.5 [89.1, 91.7]	5.8	11	3
15	resnext_29_8x64d	96.2 [95.8, 96.6]	90.1 [88.7, 91.4]	6.1	14	1
16	wrn_28_10	95.9 [95.5, 96.3]	90.1 [88.8, 91.4]	5.8	13	3
17	pyramidnet_basic_110_84	95.7 [95.3, 96.1]	89.6 [88.2, 90.9]	6.1	17	0
18	densenet_BC_100_12	95.5 [95.1, 95.9]	87.9 [86.4, 89.3]	7.6	20	-2
19	nas	95.4 [95.0, 95.8]	89.2 [87.8, 90.5]	6.2	18	1
20	wide_resnet_tf_28_10	95.0 [94.6, 95.4]	88.8 [87.4, 90.2]	6.2	19	1
21	resnet_v2_bottleneck_16	94.2 [93.7, 94.6]	86.1 [84.5, 87.6]	8.1	22	-1
22	vgg16_keras	93.6 [93.1, 94.1]	85.6 [84.0, 87.1]	8.0	23	-1
23	resnet_basic_110	93.5 [93.0, 93.9]	85.4 [83.8, 86.9]	8.1	24	-1
24	resnet_v2_basic_110	93.4 [92.9, 93.9]	86.9 [85.4, 88.3]	6.5	21	3
25	resnet_basic_56	93.3 [92.8, 93.8]	84.9 [83.2, 86.4]	8.5	28	-3
26	resnet_basic_44	93.0 [92.5, 93.5]	84.8 [83.2, 86.3]	8.2	29	-3
27	vgg_15_BN_64	93.0 [92.5, 93.5]	85.0 [83.4, 86.6]	7.9	27	0
28	resnetv2_tf_32	92.7 [92.2, 93.2]	85.1 [83.5, 86.6]	7.6	26	2
29	resnet_basic_32	92.5 [92.0, 93.0]	85.2 [83.6, 86.7]	7.3	25	4
30	cudaconvnet	88.5 [87.9, 89.2]	78.2 [76.3, 80.0]	10.3	30	0
31	random_features_256k_au	85.6 [84.9, 86.3]	73.6 [71.6, 75.5]	12.0	31	0
32	random_features_32k_aug	85.0 [84.3, 85.7]	72.2 [70.2, 74.1]	12.8	32	0
33	random_features_256k	84.2 [83.5, 84.9]	70.5 [68.4, 72.4]	13.8	33	0
34	random_features_32k	83.3 [82.6, 84.0]	68.7 [66.6, 70.7]	14.6	34	0

3.11 Supplementary material for ImageNet

In this section we provide supplementary material related to our ImageNet experiments.

3.11.0.1 Full list of models evaluated on ImageNet

The following list contains all models we evaluated on ImageNet with references and links to the corresponding source code.

1. `alexnet` [54] `https://github.com/Cadene/pretrained-models.pytorch`
2. `bninception` [42] `https://github.com/Cadene/pretrained-models.pytorch`
3. `cafferesnet101` [32] `https://github.com/Cadene/pretrained-models.pytorch`
4. `densenet121` [40] `https://github.com/Cadene/pretrained-models.pytorch`
5. `densenet161` [40]`https://github.com/Cadene/pretrained-models.pytorch`
6. `densenet169` [40] `https://github.com/Cadene/pretrained-models.pytorch`
7. `densenet201` [40] `https://github.com/Cadene/pretrained-models.pytorch`
8. `dpn107` [6] `https://github.com/Cadene/pretrained-models.pytorch`
9. `dpn131` [6] `https://github.com/Cadene/pretrained-models.pytorch`
10. `dpn68b` [6]`https://github.com/Cadene/pretrained-models.pytorch`
11. `dpn68` [6] `https://github.com/Cadene/pretrained-models.pytorch`
12. `dpn92` [6] `https://github.com/Cadene/pretrained-models.pytorch`
13. `dpn98` [6] `https://github.com/Cadene/pretrained-models.pytorch`
14. `fbresnet152` [32] `https://github.com/tensorflow/models/tree/master/research/slim/`
15. `fv_4k` [71, 9] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 16 GMM centers.
16. `fv_16k` [71, 9] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 64 GMM centers.
17. `fv_64k` [71, 9] `https://github.com/modestyachts/nondeep` FisherVector model using SIFT, local color statistic features, and 256 GMM centers.
18. `inception_resnet_v2_tf` [87] `https://github.com/tensorflow/models/tree/master/research/slim/`

19. inception_v1_tf [86] https://github.com/tensorflow/models/tree/master/research/slim/
20. inception_v2_tf [42] https://github.com/tensorflow/models/tree/master/research/slim/
21. inception_v3_tf [89] https://github.com/tensorflow/models/tree/master/research/slim/
22. inception_v3 [89] https://github.com/Cadene/pretrained-models.pytorch
23. inception_v4_tf [87] https://github.com/tensorflow/models/tree/master/research/slim/
24. inceptionresnetv2 [42] https://github.com/Cadene/pretrained-models.pytorch
25. inceptionv3 [89] https://github.com/Cadene/pretrained-models.pytorch
26. inceptionv4 [87] https://github.com/Cadene/pretrained-models.pytorch
27. mobilenet_v1_tf [38] https://github.com/tensorflow/models/tree/master/research/slim/
28. nasnet_large_tf [104] https://github.com/tensorflow/models/tree/master/research/slim/
29. nasnet_mobile_tf [104] https://github.com/tensorflow/models/tree/master/research/slim/
30. nasnetalarge [104] https://github.com/Cadene/pretrained-models.pytorch
31. nasnetamobile [104] https://github.com/Cadene/pretrained-models.pytorch
32. pnasnet5large [58] https://github.com/Cadene/pretrained-models.pytorch
33. pnasnet_large_tf [58] https://github.com/tensorflow/models/tree/master/research/slim/
34. pnasnet_mobile_tf [58] https://github.com/tensorflow/models/tree/master/research/slim/
35. polynet [103] https://github.com/Cadene/pretrained-models.pytorch
36. resnet101 [32] https://github.com/Cadene/pretrained-models.pytorch
37. resnet152 [32] https://github.com/Cadene/pretrained-models.pytorch

38. resnet18 [32] https://github.com/Cadene/pretrained-models.pytorch
39. resnet34 [32] https://github.com/Cadene/pretrained-models.pytorch
40. resnet50 [32] https://github.com/Cadene/pretrained-models.pytorch
41. resnet_v1_101_tf [32] https://github.com/tensorflow/models/tree/master/research/slim/
42. resnet_v1_152_tf [32] https://github.com/tensorflow/models/tree/master/research/slim/
43. resnet_v1_50_tf [32] https://github.com/tensorflow/models/tree/master/research/slim/
44. resnet_v2_101_tf [34] https://github.com/tensorflow/models/tree/master/research/slim/
45. resnet_v2_152_tf [34] https://github.com/tensorflow/models/tree/master/research/slim/
46. resnet_v2_50_tf [34] https://github.com/tensorflow/models/tree/master/research/slim/
47. resnext101_32x4d [96] https://github.com/Cadene/pretrained-models.pytorch
48. resnext101_64x4d [96] https://github.com/Cadene/pretrained-models.pytorch
49. se_resnet101 [39] https://github.com/Cadene/pretrained-models.pytorch
50. se_resnet152 [39] https://github.com/Cadene/pretrained-models.pytorch
51. se_resnet50 [39] https://github.com/Cadene/pretrained-models.pytorch
52. se_resnext101_32x4d [39] https://github.com/Cadene/pretrained-models.pytorch
53. se_resnext50_32x4d [39] https://github.com/Cadene/pretrained-models.pytorch
54. senet154 [39] https://github.com/Cadene/pretrained-models.pytorch
55. squeezenet1_0 [41] https://github.com/Cadene/pretrained-models.pytorch
56. squeezenet1_1 [41] https://github.com/Cadene/pretrained-models.pytorch
57. vgg11_bn [42] https://github.com/Cadene/pretrained-models.pytorch
58. vgg11 [84] https://github.com/Cadene/pretrained-models.pytorch

59. vgg13_bn [42] https://github.com/Cadene/pretrained-models.pytorch
60. vgg13 [84] https://github.com/Cadene/pretrained-models.pytorch
61. vgg16_bn [42] https://github.com/Cadene/pretrained-models.pytorch
62. vgg16 [84] https://github.com/Cadene/pretrained-models.pytorch
63. vgg19_bn [42] https://github.com/Cadene/pretrained-models.pytorch
64. vgg19 [84] https://github.com/Cadene/pretrained-models.pytorch
65. vgg_16_tf [84] https://github.com/tensorflow/models/tree/master/research/slim/
66. vgg_19_tf [84] https://github.com/tensorflow/models/tree/master/research/slim/
67. xception [8] https://github.com/Cadene/pretrained-models.pytorch

3.11.0.2 Full results tables

Tables 3.14 and 3.15 contain the detailed accuracy scores (top-1 and top-5, respectively) for the original ImageNet validation set and our main new test set MatchedFrequency. Tables 3.16 – 3.19 contain the accuracy scores for our Threshold0.7 and TopImages test sets.

Table 3.14: Top-1 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-1 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	72.2 [71.3, 73.1]	10.7	3	-2
2	pnasnet5large	82.7 [82.4, 83.1]	72.1 [71.2, 73.0]	10.7	4	-2
3	nasnet_large_tf	82.7 [82.4, 83.0]	72.2 [71.3, 73.1]	10.5	2	1
4	nasnetalarge	82.5 [82.2, 82.8]	72.2 [71.3, 73.1]	10.3	1	3
5	senet154	81.3 [81.0, 81.6]	70.5 [69.6, 71.4]	10.8	5	0
6	polynet	80.9 [80.5, 81.2]	70.3 [69.4, 71.2]	10.5	6	0
7	inception_resnet_v2_tf	80.4 [80.0, 80.7]	69.7 [68.7, 70.6]	10.7	7	0
8	inceptionresnetv2	80.3 [79.9, 80.6]	69.6 [68.7, 70.5]	10.6	8	0
9	se_resnext101_32x4d	80.2 [79.9, 80.6]	69.3 [68.4, 70.2]	10.9	9	0
10	inception_v4_tf	80.2 [79.8, 80.5]	68.8 [67.9, 69.7]	11.4	11	-1
11	inceptionv4	80.1 [79.7, 80.4]	69.1 [68.2, 70.0]	10.9	10	1
12	dpn107	79.7 [79.4, 80.1]	68.1 [67.2, 69.0]	11.7	12	0
13	dpn131	79.4 [79.1, 79.8]	67.9 [67.0, 68.8]	11.5	13	0
14	dpn92	79.4 [79.0, 79.8]	67.3 [66.3, 68.2]	12.1	17	-3
15	dpn98	79.2 [78.9, 79.6]	67.8 [66.9, 68.8]	11.4	15	0
16	se_resnext50_32x4d	79.1 [78.7, 79.4]	67.9 [66.9, 68.8]	11.2	14	2
17	resnext101_64x4d	79.0 [78.6, 79.3]	67.1 [66.2, 68.0]	11.9	20	-3
18	xception	78.8 [78.5, 79.2]	67.2 [66.2, 68.1]	11.7	18	0
19	se_resnet152	78.7 [78.3, 79.0]	67.5 [66.6, 68.5]	11.1	16	3
20	se_resnet101	78.4 [78.0, 78.8]	67.2 [66.2, 68.1]	11.2	19	1
21	resnet152	78.3 [77.9, 78.7]	67.0 [66.1, 67.9]	11.3	21	0
22	resnext101_32x4d	78.2 [77.8, 78.5]	66.2 [65.3, 67.2]	11.9	22	0
23	inception_v3_tf	78.0 [77.6, 78.3]	66.1 [65.1, 67.0]	11.9	24	-1
24	resnet_v2_152_tf	77.8 [77.4, 78.1]	66.1 [65.1, 67.0]	11.7	25	-1
25	se_resnet50	77.6 [77.3, 78.0]	66.2 [65.3, 67.2]	11.4	23	2
26	fbresnet152	77.4 [77.0, 77.8]	65.8 [64.9, 66.7]	11.6	26	0
27	resnet101	77.4 [77.0, 77.7]	65.7 [64.7, 66.6]	11.7	28	-1
28	inceptionv3	77.3 [77.0, 77.7]	65.7 [64.8, 66.7]	11.6	27	1
29	inception_v3	77.2 [76.8, 77.6]	65.4 [64.5, 66.4]	11.8	29	0
30	densenet161	77.1 [76.8, 77.5]	65.3 [64.4, 66.2]	11.8	30	0
31	dpn68b	77.0 [76.7, 77.4]	64.7 [63.7, 65.6]	12.4	32	-1
32	resnet_v2_101_tf	77.0 [76.6, 77.3]	64.6 [63.7, 65.6]	12.3	34	-2
33	densenet201	76.9 [76.5, 77.3]	64.7 [63.7, 65.6]	12.2	31	2

ImageNet Top-1 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	64.6 [63.7, 65.6]	12.2	33	1
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	64.5 [63.6, 65.5]	11.9	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	64.3 [63.4, 65.2]	11.9	36	0
37	resnet50	76.1 [75.8, 76.5]	63.3 [62.4, 64.3]	12.8	39	-2
38	dpn68	75.9 [75.5, 76.2]	63.4 [62.5, 64.4]	12.4	38	0
39	densenet169	75.6 [75.2, 76.0]	63.9 [62.9, 64.8]	11.7	37	2
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	62.7 [61.8, 63.7]	12.9	40	0
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	62.6 [61.6, 63.5]	12.6	41	0
42	densenet121	74.4 [74.0, 74.8]	62.2 [61.3, 63.2]	12.2	42	0
43	vgg19_bn	74.2 [73.8, 74.6]	61.9 [60.9, 62.8]	12.3	44	-1
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	60.9 [59.9, 61.8]	13.3	48	-4
45	nasnetamobile	74.1 [73.7, 74.5]	61.6 [60.6, 62.5]	12.5	45	0
46	inception_v2_tf	74.0 [73.6, 74.4]	61.2 [60.2, 62.2]	12.8	46	0
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	60.8 [59.8, 61.7]	13.2	50	-3
48	bninception	73.5 [73.1, 73.9]	62.1 [61.2, 63.1]	11.4	43	5
49	vgg16_bn	73.4 [73.0, 73.7]	60.8 [59.8, 61.7]	12.6	49	0
50	resnet34	73.3 [72.9, 73.7]	61.2 [60.2, 62.2]	12.1	47	3
51	vgg19	72.4 [72.0, 72.8]	59.7 [58.7, 60.7]	12.7	51	0
52	vgg16	71.6 [71.2, 72.0]	58.8 [57.9, 59.8]	12.8	53	-1
53	vgg13_bn	71.6 [71.2, 72.0]	59.0 [58.0, 59.9]	12.6	52	1
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	57.4 [56.4, 58.4]	13.6	56	-2
55	vgg_19_tf	71.0 [70.6, 71.4]	58.6 [57.7, 59.6]	12.4	54	1
56	vgg_16_tf	70.9 [70.5, 71.3]	58.4 [57.4, 59.3]	12.5	55	1
57	vgg11_bn	70.4 [70.0, 70.8]	57.4 [56.4, 58.4]	13.0	57	0
58	vgg13	69.9 [69.5, 70.3]	57.1 [56.2, 58.1]	12.8	59	-1
59	inception_v1_tf	69.8 [69.4, 70.2]	56.6 [55.7, 57.6]	13.1	60	-1
60	resnet18	69.8 [69.4, 70.2]	57.2 [56.2, 58.2]	12.6	58	2
61	vgg11	69.0 [68.6, 69.4]	55.8 [54.8, 56.8]	13.2	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	45.3 [44.4, 46.3]	12.8	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	45.0 [44.0, 46.0]	13.1	63	0
64	alexnet	56.5 [56.1, 57.0]	44.0 [43.0, 45.0]	12.5	64	0
65	fv_64k	35.1 [34.7, 35.5]	24.1 [23.2, 24.9]	11.0	65	0
66	fv_16k	28.3 [27.9, 28.7]	19.2 [18.5, 20.0]	9.1	66	0
67	fv_4k	21.2 [20.8, 21.5]	15.0 [14.3, 15.7]	6.2	67	0

Table 3.15: Top-5 model accuracy on the original ImageNet validation set and our new test set MatchedFrequency. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-5 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	96.2 [96.0, 96.3]	90.1 [89.5, 90.7]	6.1	3	-2
2	nasnet_large_tf	96.2 [96.0, 96.3]	90.1 [89.5, 90.6]	6.1	4	-2
3	nasnetalarge	96.0 [95.8, 96.2]	90.4 [89.8, 91.0]	5.6	1	2
4	pnasnet5large	96.0 [95.8, 96.2]	90.2 [89.6, 90.8]	5.8	2	2
5	polynet	95.6 [95.4, 95.7]	89.1 [88.5, 89.7]	6.4	5	0
6	senet154	95.5 [95.3, 95.7]	89.0 [88.4, 89.6]	6.5	6	0
7	inception_resnet_v2_tf	95.2 [95.1, 95.4]	88.4 [87.7, 89.0]	6.9	9	-2
8	inception_v4_tf	95.2 [95.0, 95.4]	88.3 [87.6, 88.9]	6.9	10	-2
9	inceptionresnetv2	95.1 [94.9, 95.3]	88.5 [87.8, 89.1]	6.7	8	1
10	se_resnext101_32x4d	95.0 [94.8, 95.2]	88.0 [87.4, 88.7]	7.0	11	-1
11	inceptionv4	94.9 [94.7, 95.1]	88.7 [88.1, 89.3]	6.2	7	4
12	dpn107	94.7 [94.5, 94.9]	87.6 [86.9, 88.2]	7.1	13	-1
13	dpn92	94.6 [94.4, 94.8]	87.2 [86.5, 87.8]	7.5	17	-4
14	dpn131	94.6 [94.4, 94.8]	87.0 [86.3, 87.7]	7.6	19	-5
15	dpn98	94.5 [94.3, 94.7]	87.2 [86.5, 87.8]	7.3	16	-1
16	se_resnext50_32x4d	94.4 [94.2, 94.6]	87.6 [87.0, 88.3]	6.8	12	4
17	se_resnet152	94.4 [94.2, 94.6]	87.4 [86.7, 88.0]	7.0	15	2
18	xception	94.3 [94.1, 94.5]	87.0 [86.3, 87.7]	7.3	20	-2
19	se_resnet101	94.3 [94.1, 94.5]	87.1 [86.4, 87.7]	7.2	18	1
20	resnext101_64x4d	94.3 [94.0, 94.5]	86.9 [86.2, 87.5]	7.4	22	-2
21	resnet_v2_152_tf	94.1 [93.9, 94.3]	86.9 [86.2, 87.5]	7.2	21	0
22	resnet152	94.0 [93.8, 94.3]	87.6 [86.9, 88.2]	6.5	14	8
23	inception_v3_tf	93.9 [93.7, 94.1]	86.4 [85.7, 87.0]	7.6	23	0
24	resnext101_32x4d	93.9 [93.7, 94.1]	86.2 [85.5, 86.8]	7.7	25	-1
25	se_resnet50	93.8 [93.5, 94.0]	86.3 [85.6, 87.0]	7.4	24	1
26	resnet_v2_101_tf	93.7 [93.5, 93.9]	86.1 [85.4, 86.8]	7.6	27	-1
27	fbresnet152	93.6 [93.4, 93.8]	86.1 [85.4, 86.7]	7.5	28	-1
28	dpn68b	93.6 [93.4, 93.8]	85.3 [84.6, 86.0]	8.3	33	-5
29	densenet161	93.6 [93.3, 93.8]	86.1 [85.4, 86.8]	7.4	26	3
30	resnet101	93.5 [93.3, 93.8]	86.0 [85.3, 86.7]	7.6	30	0
31	inception_v3	93.5 [93.3, 93.7]	85.9 [85.2, 86.6]	7.6	31	0
32	inceptionv3	93.4 [93.2, 93.6]	86.1 [85.4, 86.7]	7.4	29	3
33	densenet201	93.4 [93.1, 93.6]	85.3 [84.6, 86.0]	8.1	34	-1

ImageNet Top-5 MatchedFrequency						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	93.2 [92.9, 93.4]	85.4 [84.6, 86.0]	7.8	32	2
35	resnet_v1_101_tf	92.9 [92.7, 93.1]	85.2 [84.5, 85.9]	7.7	35	0
36	resnet50	92.9 [92.6, 93.1]	84.7 [83.9, 85.4]	8.2	38	-2
37	resnet_v2_50_tf	92.8 [92.6, 93.1]	84.4 [83.6, 85.1]	8.5	40	-3
38	densenet169	92.8 [92.6, 93.0]	84.7 [84.0, 85.4]	8.1	37	1
39	dpn68	92.8 [92.5, 93.0]	84.6 [83.9, 85.3]	8.2	39	0
40	cafferesnet101	92.8 [92.5, 93.0]	84.9 [84.1, 85.6]	7.9	36	4
41	resnet_v1_50_tf	92.2 [92.0, 92.4]	84.1 [83.4, 84.8]	8.1	41	0
42	densenet121	92.0 [91.7, 92.2]	83.8 [83.1, 84.5]	8.2	42	0
43	pnasnet_mobile_tf	91.9 [91.6, 92.1]	83.1 [82.4, 83.8]	8.8	46	-3
44	vgg19_bn	91.8 [91.6, 92.1]	83.5 [82.7, 84.2]	8.4	43	1
45	inception_v2_tf	91.8 [91.5, 92.0]	83.1 [82.3, 83.8]	8.7	47	-2
46	nasnetamobile	91.7 [91.5, 92.0]	83.4 [82.6, 84.1]	8.4	45	1
47	nasnet_mobile_tf	91.6 [91.3, 91.8]	82.2 [81.4, 82.9]	9.4	50	-3
48	bninception	91.6 [91.3, 91.8]	83.4 [82.7, 84.2]	8.1	44	4
49	vgg16_bn	91.5 [91.3, 91.8]	83.0 [82.2, 83.7]	8.6	48	1
50	resnet34	91.4 [91.2, 91.7]	82.7 [82.0, 83.5]	8.7	49	1
51	vgg19	90.9 [90.6, 91.1]	81.5 [80.7, 82.2]	9.4	52	-1
52	vgg16	90.4 [90.1, 90.6]	81.7 [80.9, 82.4]	8.7	51	1
53	vgg13_bn	90.4 [90.1, 90.6]	81.1 [80.3, 81.9]	9.3	53	0
54	mobilenet_v1_tf	90.0 [89.7, 90.2]	79.4 [78.6, 80.1]	10.6	60	-6
56	vgg_19_tf	89.8 [89.6, 90.1]	80.7 [79.9, 81.4]	9.2	54	2
55	vgg_16_tf	89.8 [89.6, 90.1]	80.5 [79.7, 81.3]	9.3	55	0
57	vgg11_bn	89.8 [89.5, 90.1]	80.0 [79.2, 80.8]	9.8	58	-1
58	inception_v1_tf	89.6 [89.4, 89.9]	80.1 [79.3, 80.9]	9.5	57	1
59	vgg13	89.2 [89.0, 89.5]	79.5 [78.7, 80.3]	9.7	59	0
60	resnet18	89.1 [88.8, 89.3]	80.2 [79.4, 81.0]	8.9	56	4
61	vgg11	88.6 [88.3, 88.9]	78.8 [78.0, 79.6]	9.8	61	0
62	squeezenet1_1	80.6 [80.3, 81.0]	69.0 [68.1, 69.9]	11.6	62	0
63	squeezenet1_0	80.4 [80.1, 80.8]	68.5 [67.6, 69.4]	11.9	63	0
64	alexnet	79.1 [78.7, 79.4]	67.4 [66.5, 68.3]	11.7	64	0
65	fv_64k	55.7 [55.3, 56.2]	42.6 [41.6, 43.6]	13.2	65	0
66	fv_16k	49.9 [49.5, 50.4]	37.5 [36.6, 38.5]	12.4	66	0
67	fv_4k	41.3 [40.8, 41.7]	31.0 [30.1, 31.9]	10.3	67	0

Table 3.16: Top-1 model accuracy on the original ImageNet validation set and our new test set Threshold0.7. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, $\Delta\text{Rank} = -2$ means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-1 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	80.2 [79.4, 80.9]	2.7	2	-1
2	pnasnet5large	82.7 [82.4, 83.1]	80.3 [79.5, 81.1]	2.4	1	1
3	nasnet_large_tf	82.7 [82.4, 83.0]	80.1 [79.3, 80.9]	2.6	3	0
4	nasnetalarge	82.5 [82.2, 82.8]	80.0 [79.2, 80.8]	2.5	4	0
5	senet154	81.3 [81.0, 81.6]	78.7 [77.8, 79.5]	2.6	5	0
6	polynet	80.9 [80.5, 81.2]	78.5 [77.7, 79.3]	2.3	6	0
7	inception_resnet_v2_tf	80.4 [80.0, 80.7]	77.9 [77.1, 78.7]	2.5	8	-1
8	inceptionresnetv2	80.3 [79.9, 80.6]	78.0 [77.2, 78.8]	2.3	7	1
9	se_resnext101_32x4d	80.2 [79.9, 80.6]	77.6 [76.8, 78.5]	2.6	11	-2
10	inception_v4_tf	80.2 [79.8, 80.5]	77.8 [77.0, 78.6]	2.4	10	0
11	inceptionv4	80.1 [79.7, 80.4]	77.9 [77.0, 78.7]	2.2	9	2
12	dpn107	79.7 [79.4, 80.1]	76.6 [75.8, 77.5]	3.1	12	0
13	dpn131	79.4 [79.1, 79.8]	76.6 [75.7, 77.4]	2.9	13	0
14	dpn92	79.4 [79.0, 79.8]	76.3 [75.5, 77.1]	3.1	17	-3
15	dpn98	79.2 [78.9, 79.6]	76.3 [75.5, 77.2]	2.9	16	-1
16	se_resnext50_32x4d	79.1 [78.7, 79.4]	76.5 [75.7, 77.3]	2.6	14	2
17	resnext101_64x4d	79.0 [78.6, 79.3]	75.6 [74.7, 76.4]	3.4	20	-3
18	xception	78.8 [78.5, 79.2]	76.4 [75.5, 77.2]	2.5	15	3
19	se_resnet152	78.7 [78.3, 79.0]	76.1 [75.3, 76.9]	2.5	18	1
20	se_resnet101	78.4 [78.0, 78.8]	75.8 [75.0, 76.7]	2.6	19	1
21	resnet152	78.3 [77.9, 78.7]	75.3 [74.5, 76.2]	3.0	22	-1
22	resnext101_32x4d	78.2 [77.8, 78.5]	75.4 [74.5, 76.2]	2.8	21	1
23	inception_v3_tf	78.0 [77.6, 78.3]	75.0 [74.2, 75.9]	2.9	24	-1
24	resnet_v2_152_tf	77.8 [77.4, 78.1]	75.2 [74.4, 76.1]	2.6	23	1
25	se_resnet50	77.6 [77.3, 78.0]	74.2 [73.3, 75.1]	3.4	30	-5
26	fbresnet152	77.4 [77.0, 77.8]	74.8 [74.0, 75.7]	2.6	25	1
27	resnet101	77.4 [77.0, 77.7]	74.5 [73.6, 75.3]	2.9	29	-2
28	inceptionv3	77.3 [77.0, 77.7]	74.5 [73.6, 75.4]	2.8	28	0
29	inception_v3	77.2 [76.8, 77.6]	74.7 [73.8, 75.6]	2.5	26	3
30	densenet161	77.1 [76.8, 77.5]	74.6 [73.7, 75.4]	2.6	27	3
31	dpn68b	77.0 [76.7, 77.4]	73.8 [72.9, 74.7]	3.2	33	-2
32	resnet_v2_101_tf	77.0 [76.6, 77.3]	74.0 [73.1, 74.8]	3.0	31	1
33	densenet201	76.9 [76.5, 77.3]	73.9 [73.1, 74.8]	3.0	32	1

ImageNet Top-1 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	73.7 [72.9, 74.6]	3.1	34	0
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	73.4 [72.5, 74.2]	3.0	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	72.9 [72.0, 73.7]	3.3	37	-1
37	resnet50	76.1 [75.8, 76.5]	72.7 [71.8, 73.6]	3.4	38	-1
38	dpn68	75.9 [75.5, 76.2]	73.0 [72.1, 73.8]	2.9	36	2
39	densenet169	75.6 [75.2, 76.0]	72.3 [71.4, 73.1]	3.3	40	-1
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	72.3 [71.4, 73.2]	3.3	39	1
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	71.9 [71.0, 72.8]	3.3	41	0
42	densenet121	74.4 [74.0, 74.8]	70.5 [69.6, 71.4]	3.9	47	-5
43	vgg19_bn	74.2 [73.8, 74.6]	71.4 [70.5, 72.3]	2.8	42	1
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	70.6 [69.7, 71.5]	3.6	46	-2
45	nasnetamobile	74.1 [73.7, 74.5]	70.9 [70.0, 71.8]	3.2	45	0
46	inception_v2_tf	74.0 [73.6, 74.4]	71.1 [70.2, 72.0]	2.9	44	2
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	70.0 [69.0, 70.8]	4.0	50	-3
48	bninception	73.5 [73.1, 73.9]	71.3 [70.4, 72.2]	2.2	43	5
49	vgg16_bn	73.4 [73.0, 73.7]	70.2 [69.3, 71.1]	3.1	48	1
50	resnet34	73.3 [72.9, 73.7]	70.2 [69.2, 71.0]	3.2	49	1
51	vgg19	72.4 [72.0, 72.8]	68.7 [67.8, 69.6]	3.7	51	0
52	vgg16	71.6 [71.2, 72.0]	68.0 [67.0, 68.9]	3.6	52	0
53	vgg13_bn	71.6 [71.2, 72.0]	67.3 [66.4, 68.2]	4.3	55	-2
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	66.1 [65.2, 67.0]	4.9	59	-5
55	vgg_19_tf	71.0 [70.6, 71.4]	67.4 [66.5, 68.3]	3.6	54	1
56	vgg_16_tf	70.9 [70.5, 71.3]	67.6 [66.7, 68.5]	3.3	53	3
57	vgg11_bn	70.4 [70.0, 70.8]	66.4 [65.5, 67.3]	4.0	58	-1
58	vgg13	69.9 [69.5, 70.3]	66.0 [65.0, 66.9]	4.0	60	-2
59	inception_v1_tf	69.8 [69.4, 70.2]	66.4 [65.5, 67.4]	3.3	57	2
60	resnet18	69.8 [69.4, 70.2]	66.6 [65.7, 67.5]	3.2	56	4
61	vgg11	69.0 [68.6, 69.4]	64.6 [63.7, 65.6]	4.4	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	54.4 [53.4, 55.4]	3.8	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	53.4 [52.4, 54.4]	4.7	63	0
64	alexnet	56.5 [56.1, 57.0]	51.3 [50.3, 52.3]	5.2	64	0
65	fv_64k	35.1 [34.7, 35.5]	29.1 [28.2, 30.0]	6.0	65	0
66	fv_16k	28.3 [27.9, 28.7]	23.4 [22.5, 24.2]	5.0	66	0
67	fv_4k	21.2 [20.8, 21.5]	17.8 [17.0, 18.5]	3.4	67	0

Table 3.17: Top-5 model accuracy on the original ImageNet validation set and our new test set Threshold0.7. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-5 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	96.2 [96.0, 96.3]	95.6 [95.2, 96.0]	0.6	2	-1
2	nasnet_large_tf	96.2 [96.0, 96.3]	95.7 [95.2, 96.0]	0.5	1	1
3	nasnetalarge	96.0 [95.8, 96.2]	95.3 [94.9, 95.8]	0.7	4	-1
4	pnasnet5large	96.0 [95.8, 96.2]	95.5 [95.0, 95.9]	0.5	3	1
5	polynet	95.6 [95.4, 95.7]	94.9 [94.4, 95.3]	0.7	5	0
6	senet154	95.5 [95.3, 95.7]	94.8 [94.3, 95.2]	0.7	6	0
7	inception_resnet_v2_tf	95.2 [95.1, 95.4]	94.7 [94.2, 95.1]	0.6	7	0
8	inception_v4_tf	95.2 [95.0, 95.4]	94.4 [94.0, 94.9]	0.8	9	-1
9	inceptionresnetv2	95.1 [94.9, 95.3]	94.5 [94.1, 95.0]	0.6	8	1
10	se_resnext101_32x4d	95.0 [94.8, 95.2]	94.3 [93.8, 94.7]	0.7	11	-1
11	inceptionv4	94.9 [94.7, 95.1]	94.3 [93.8, 94.7]	0.6	10	1
12	dpn107	94.7 [94.5, 94.9]	93.7 [93.2, 94.2]	1.0	12	0
13	dpn92	94.6 [94.4, 94.8]	93.7 [93.2, 94.2]	0.9	14	-1
14	dpn131	94.6 [94.4, 94.8]	93.5 [92.9, 93.9]	1.1	20	-6
15	dpn98	94.5 [94.3, 94.7]	93.6 [93.1, 94.1]	0.9	17	-2
16	se_resnext50_32x4d	94.4 [94.2, 94.6]	93.6 [93.1, 94.1]	0.8	16	0
17	se_resnet152	94.4 [94.2, 94.6]	93.7 [93.2, 94.2]	0.7	13	4
18	xception	94.3 [94.1, 94.5]	93.6 [93.1, 94.1]	0.7	18	0
19	se_resnet101	94.3 [94.1, 94.5]	93.6 [93.1, 94.0]	0.7	19	0
20	resnext101_64x4d	94.3 [94.0, 94.5]	93.3 [92.8, 93.8]	0.9	22	-2
21	resnet_v2_152_tf	94.1 [93.9, 94.3]	93.4 [92.9, 93.9]	0.7	21	0
22	resnet152	94.0 [93.8, 94.3]	93.7 [93.2, 94.2]	0.4	15	7
23	inception_v3_tf	93.9 [93.7, 94.1]	92.8 [92.3, 93.3]	1.1	25	-2
24	resnext101_32x4d	93.9 [93.7, 94.1]	92.7 [92.2, 93.2]	1.2	28	-4
25	se_resnet50	93.8 [93.5, 94.0]	93.0 [92.4, 93.5]	0.8	24	1
26	resnet_v2_101_tf	93.7 [93.5, 93.9]	93.2 [92.7, 93.7]	0.5	23	3
27	fbresnet152	93.6 [93.4, 93.8]	92.7 [92.1, 93.2]	0.9	29	-2
28	dpn68b	93.6 [93.4, 93.8]	92.7 [92.1, 93.2]	0.9	31	-3
29	densenet161	93.6 [93.3, 93.8]	92.8 [92.3, 93.3]	0.8	26	3
30	resnet101	93.5 [93.3, 93.8]	92.8 [92.3, 93.3]	0.8	27	3
31	inception_v3	93.5 [93.3, 93.7]	92.7 [92.1, 93.2]	0.9	30	1
32	inceptionv3	93.4 [93.2, 93.6]	92.6 [92.1, 93.1]	0.8	32	0
33	densenet201	93.4 [93.1, 93.6]	92.4 [91.9, 92.9]	1.0	33	0

ImageNet Top-5 Threshold0.7						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	93.2 [92.9, 93.4]	92.2 [91.7, 92.7]	1.0	34	0
35	resnet_v1_101_tf	92.9 [92.7, 93.1]	92.0 [91.5, 92.5]	0.9	36	-1
36	resnet50	92.9 [92.6, 93.1]	92.0 [91.5, 92.5]	0.9	37	-1
37	resnet_v2_50_tf	92.8 [92.6, 93.1]	91.9 [91.4, 92.5]	0.9	38	-1
38	densenet169	92.8 [92.6, 93.0]	91.9 [91.4, 92.4]	0.9	39	-1
39	dpn68	92.8 [92.5, 93.0]	92.1 [91.5, 92.6]	0.7	35	4
40	cafferesnet101	92.8 [92.5, 93.0]	91.6 [91.1, 92.2]	1.1	40	0
41	resnet_v1_50_tf	92.2 [92.0, 92.4]	91.1 [90.6, 91.7]	1.0	41	0
42	densenet121	92.0 [91.7, 92.2]	91.1 [90.5, 91.6]	0.9	42	0
43	pnasnet_mobile_tf	91.9 [91.6, 92.1]	90.7 [90.1, 91.3]	1.1	47	-4
44	vgg19_bn	91.8 [91.6, 92.1]	91.0 [90.4, 91.5]	0.9	44	0
45	inception_v2_tf	91.8 [91.5, 92.0]	91.0 [90.5, 91.6]	0.7	43	2
46	nasnetamobile	91.7 [91.5, 92.0]	90.9 [90.3, 91.4]	0.9	46	0
47	nasnet_mobile_tf	91.6 [91.3, 91.8]	90.1 [89.5, 90.7]	1.4	50	-3
48	bninception	91.6 [91.3, 91.8]	90.9 [90.3, 91.5]	0.7	45	3
49	vgg16_bn	91.5 [91.3, 91.8]	90.4 [89.8, 90.9]	1.1	49	0
50	resnet34	91.4 [91.2, 91.7]	90.5 [89.9, 91.0]	1.0	48	2
51	vgg19	90.9 [90.6, 91.1]	89.7 [89.1, 90.3]	1.2	51	0
52	vgg16	90.4 [90.1, 90.6]	88.8 [88.1, 89.4]	1.6	53	-1
53	vgg13_bn	90.4 [90.1, 90.6]	89.0 [88.3, 89.6]	1.4	52	1
54	mobilenet_v1_tf	90.0 [89.7, 90.2]	87.6 [86.9, 88.2]	2.4	60	-6
56	vgg_19_tf	89.8 [89.6, 90.1]	88.5 [87.8, 89.1]	1.4	55	1
55	vgg_16_tf	89.8 [89.6, 90.1]	88.6 [87.9, 89.2]	1.3	54	1
57	vgg11_bn	89.8 [89.5, 90.1]	88.3 [87.6, 88.9]	1.5	56	1
58	inception_v1_tf	89.6 [89.4, 89.9]	88.1 [87.4, 88.7]	1.5	57	1
59	vgg13	89.2 [89.0, 89.5]	87.6 [86.9, 88.2]	1.6	59	0
60	resnet18	89.1 [88.8, 89.3]	88.1 [87.4, 88.7]	1.0	58	2
61	vgg11	88.6 [88.3, 88.9]	86.9 [86.2, 87.5]	1.7	61	0
62	squeezenet1_1	80.6 [80.3, 81.0]	78.0 [77.2, 78.8]	2.6	62	0
63	squeezenet1_0	80.4 [80.1, 80.8]	77.7 [76.9, 78.5]	2.7	63	0
64	alexnet	79.1 [78.7, 79.4]	75.9 [75.0, 76.7]	3.2	64	0
65	fv_64k	55.7 [55.3, 56.2]	49.8 [48.8, 50.7]	6.0	65	0
66	fv_16k	49.9 [49.5, 50.4]	44.2 [43.2, 45.2]	5.7	66	0
67	fv_4k	41.3 [40.8, 41.7]	36.5 [35.6, 37.5]	4.8	67	0

Table 3.18: Top-1 model accuracy on the original ImageNet validation set and our new test set TopImages. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-1 TopImages						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	83.9 [83.2, 84.6]	-1.0	3	-2
2	pnasnet5large	82.7 [82.4, 83.1]	83.9 [83.1, 84.6]	-1.1	4	-2
3	nasnet_large_tf	82.7 [82.4, 83.0]	84.0 [83.3, 84.7]	-1.3	2	1
4	nasnetalarge	82.5 [82.2, 82.8]	84.2 [83.4, 84.9]	-1.7	1	3
5	senet154	81.3 [81.0, 81.6]	82.8 [82.1, 83.6]	-1.5	6	-1
6	polynet	80.9 [80.5, 81.2]	83.0 [82.2, 83.7]	-2.1	5	1
7	inception_resnet_v2_tf	80.4 [80.0, 80.7]	82.5 [81.7, 83.2]	-2.1	8	-1
8	inceptionresnetv2	80.3 [79.9, 80.6]	82.8 [82.0, 83.5]	-2.5	7	1
9	se_resnext101_32x4d	80.2 [79.9, 80.6]	82.2 [81.5, 83.0]	-2.0	11	-2
10	inception_v4_tf	80.2 [79.8, 80.5]	82.3 [81.5, 83.0]	-2.1	9	1
11	inceptionv4	80.1 [79.7, 80.4]	82.3 [81.5, 83.0]	-2.2	10	1
12	dpn107	79.7 [79.4, 80.1]	81.4 [80.6, 82.1]	-1.6	13	-1
13	dpn131	79.4 [79.1, 79.8]	81.3 [80.5, 82.1]	-1.9	15	-2
14	dpn92	79.4 [79.0, 79.8]	81.2 [80.5, 82.0]	-1.8	16	-2
15	dpn98	79.2 [78.9, 79.6]	81.5 [80.7, 82.3]	-2.3	12	3
16	se_resnext50_32x4d	79.1 [78.7, 79.4]	81.4 [80.6, 82.1]	-2.3	14	2
17	resnext101_64x4d	79.0 [78.6, 79.3]	80.3 [79.5, 81.0]	-1.3	22	-5
18	xception	78.8 [78.5, 79.2]	81.0 [80.2, 81.8]	-2.2	18	0
19	se_resnet152	78.7 [78.3, 79.0]	81.0 [80.3, 81.8]	-2.4	17	2
20	se_resnet101	78.4 [78.0, 78.8]	80.5 [79.7, 81.3]	-2.1	19	1
21	resnet152	78.3 [77.9, 78.7]	80.3 [79.5, 81.1]	-2.0	21	0
22	resnext101_32x4d	78.2 [77.8, 78.5]	79.9 [79.1, 80.6]	-1.7	26	-4
23	inception_v3_tf	78.0 [77.6, 78.3]	80.1 [79.3, 80.9]	-2.1	23	0
24	resnet_v2_152_tf	77.8 [77.4, 78.1]	80.3 [79.5, 81.1]	-2.6	20	4
25	se_resnet50	77.6 [77.3, 78.0]	79.4 [78.6, 80.2]	-1.8	31	-6
26	fbresnet152	77.4 [77.0, 77.8]	80.1 [79.3, 80.9]	-2.7	24	2
27	resnet101	77.4 [77.0, 77.7]	79.0 [78.2, 79.8]	-1.7	32	-5
28	inceptionv3	77.3 [77.0, 77.7]	79.6 [78.8, 80.4]	-2.3	27	1
29	inception_v3	77.2 [76.8, 77.6]	79.6 [78.8, 80.4]	-2.4	28	1
30	densenet161	77.1 [76.8, 77.5]	79.5 [78.7, 80.3]	-2.4	29	1
31	dpn68b	77.0 [76.7, 77.4]	79.4 [78.6, 80.2]	-2.4	30	1
32	resnet_v2_101_tf	77.0 [76.6, 77.3]	80.1 [79.3, 80.8]	-3.1	25	7
33	densenet201	76.9 [76.5, 77.3]	79.0 [78.1, 79.7]	-2.1	34	-1

ImageNet Top-1 TopImages						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	76.8 [76.4, 77.2]	79.0 [78.2, 79.8]	-2.2	33	1
35	resnet_v1_101_tf	76.4 [76.0, 76.8]	78.6 [77.8, 79.4]	-2.2	35	0
36	cafferesnet101	76.2 [75.8, 76.6]	78.3 [77.4, 79.1]	-2.1	37	-1
37	resnet50	76.1 [75.8, 76.5]	78.1 [77.3, 78.9]	-2.0	38	-1
38	dpn68	75.9 [75.5, 76.2]	78.4 [77.6, 79.2]	-2.6	36	2
39	densenet169	75.6 [75.2, 76.0]	78.0 [77.2, 78.8]	-2.4	39	0
40	resnet_v2_50_tf	75.6 [75.2, 76.0]	78.0 [77.2, 78.8]	-2.4	40	0
41	resnet_v1_50_tf	75.2 [74.8, 75.6]	77.0 [76.2, 77.9]	-1.8	41	0
42	densenet121	74.4 [74.0, 74.8]	76.8 [75.9, 77.6]	-2.3	45	-3
43	vgg19_bn	74.2 [73.8, 74.6]	76.6 [75.7, 77.4]	-2.3	46	-3
44	pnasnet_mobile_tf	74.1 [73.8, 74.5]	76.8 [76.0, 77.6]	-2.7	44	0
45	nasnetamobile	74.1 [73.7, 74.5]	76.4 [75.5, 77.2]	-2.3	47	-2
46	inception_v2_tf	74.0 [73.6, 74.4]	77.0 [76.1, 77.8]	-3.0	43	3
47	nasnet_mobile_tf	74.0 [73.6, 74.4]	76.0 [75.1, 76.8]	-2.0	49	-2
48	bninception	73.5 [73.1, 73.9]	77.0 [76.1, 77.8]	-3.4	42	6
49	vgg16_bn	73.4 [73.0, 73.7]	75.9 [75.1, 76.8]	-2.6	50	-1
50	resnet34	73.3 [72.9, 73.7]	76.3 [75.4, 77.1]	-3.0	48	2
51	vgg19	72.4 [72.0, 72.8]	74.2 [73.3, 75.0]	-1.8	51	0
52	vgg16	71.6 [71.2, 72.0]	73.9 [73.0, 74.7]	-2.3	52	0
53	vgg13_bn	71.6 [71.2, 72.0]	73.5 [72.7, 74.4]	-1.9	55	-2
54	mobilenet_v1_tf	71.0 [70.6, 71.4]	72.4 [71.5, 73.3]	-1.4	59	-5
55	vgg_19_tf	71.0 [70.6, 71.4]	73.6 [72.7, 74.5]	-2.6	53	2
56	vgg_16_tf	70.9 [70.5, 71.3]	73.5 [72.7, 74.4]	-2.6	54	2
57	vgg11_bn	70.4 [70.0, 70.8]	73.0 [72.1, 73.8]	-2.6	58	-1
58	vgg13	69.9 [69.5, 70.3]	72.0 [71.1, 72.9]	-2.1	60	-2
59	inception_v1_tf	69.8 [69.4, 70.2]	73.1 [72.2, 73.9]	-3.3	56	3
60	resnet18	69.8 [69.4, 70.2]	73.0 [72.2, 73.9]	-3.3	57	3
61	vgg11	69.0 [68.6, 69.4]	70.8 [69.9, 71.7]	-1.8	61	0
62	squeezenet1_1	58.2 [57.7, 58.6]	61.7 [60.7, 62.6]	-3.5	62	0
63	squeezenet1_0	58.1 [57.7, 58.5]	60.7 [59.7, 61.7]	-2.6	63	0
64	alexnet	56.5 [56.1, 57.0]	58.2 [57.2, 59.1]	-1.7	64	0
65	fv_64k	35.1 [34.7, 35.5]	34.2 [33.3, 35.2]	0.8	65	0
66	fv_16k	28.3 [27.9, 28.7]	27.4 [26.6, 28.3]	0.9	66	0
67	fv_4k	21.2 [20.8, 21.5]	21.1 [20.3, 21.9]	0.1	67	0

Table 3.19: Top-5 model accuracy on the original ImageNet validation set and our new test set TopImages. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means that a model dropped by two places on the new test set compared to the original test set. The confidence intervals are 95% Clopper-Pearson intervals. References for the models can be found in Section 3.11.0.1. The second part of the table can be found on the following page.

ImageNet Top-5 TopImages

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	96.2 [96.0, 96.3]	97.2 [96.9, 97.5]	-1.0	2	-1
2	nasnet_large_tf	96.2 [96.0, 96.3]	97.2 [96.9, 97.5]	-1.0	1	1
3	nasnetalarge	96.0 [95.8, 96.2]	97.1 [96.7, 97.4]	-1.1	3	0
4	pnasnet5large	96.0 [95.8, 96.2]	96.9 [96.6, 97.2]	-0.9	4	0
5	polynet	95.6 [95.4, 95.7]	96.8 [96.4, 97.1]	-1.2	5	0
6	senet154	95.5 [95.3, 95.7]	96.6 [96.2, 97.0]	-1.1	8	-2
7	inception_resnet_v2_tf	95.2 [95.1, 95.4]	96.8 [96.4, 97.1]	-1.5	6	1
8	inception_v4_tf	95.2 [95.0, 95.4]	96.5 [96.1, 96.9]	-1.3	9	-1
9	inceptionresnetv2	95.1 [94.9, 95.3]	96.7 [96.3, 97.0]	-1.5	7	2
10	se_resnext101_32x4d	95.0 [94.8, 95.2]	96.2 [95.8, 96.6]	-1.2	11	-1
11	inceptionv4	94.9 [94.7, 95.1]	96.4 [96.0, 96.7]	-1.5	10	1
12	dpn107	94.7 [94.5, 94.9]	96.0 [95.6, 96.4]	-1.4	13	-1
13	dpn92	94.6 [94.4, 94.8]	95.9 [95.5, 96.3]	-1.3	17	-4
14	dpn131	94.6 [94.4, 94.8]	96.0 [95.6, 96.4]	-1.5	14	0
15	dpn98	94.5 [94.3, 94.7]	96.0 [95.6, 96.4]	-1.5	15	0
16	se_resnext50_32x4d	94.4 [94.2, 94.6]	95.9 [95.5, 96.3]	-1.5	18	-2
17	se_resnet152	94.4 [94.2, 94.6]	95.9 [95.5, 96.3]	-1.5	19	-2
18	xception	94.3 [94.1, 94.5]	95.9 [95.5, 96.3]	-1.6	20	-2
19	se_resnet101	94.3 [94.1, 94.5]	95.9 [95.5, 96.3]	-1.6	21	-2
20	resnext101_64x4d	94.3 [94.0, 94.5]	95.7 [95.3, 96.1]	-1.5	23	-3
21	resnet_v2_152_tf	94.1 [93.9, 94.3]	96.0 [95.6, 96.3]	-1.9	16	5
22	resnet152	94.0 [93.8, 94.3]	96.2 [95.8, 96.5]	-2.1	12	10
23	inception_v3_tf	93.9 [93.7, 94.1]	95.5 [95.1, 95.9]	-1.5	25	-2
24	resnext101_32x4d	93.9 [93.7, 94.1]	95.2 [94.8, 95.6]	-1.3	31	-7
25	se_resnet50	93.8 [93.5, 94.0]	95.5 [95.1, 95.9]	-1.8	24	1
26	resnet_v2_101_tf	93.7 [93.5, 93.9]	95.8 [95.4, 96.2]	-2.1	22	4
27	fbresnet152	93.6 [93.4, 93.8]	95.2 [94.8, 95.7]	-1.7	28	-1
28	dpn68b	93.6 [93.4, 93.8]	95.2 [94.8, 95.6]	-1.6	32	-4
29	densenet161	93.6 [93.3, 93.8]	95.2 [94.8, 95.6]	-1.7	29	0
30	resnet101	93.5 [93.3, 93.8]	95.4 [95.0, 95.8]	-1.9	26	4
31	inception_v3	93.5 [93.3, 93.7]	95.1 [94.7, 95.5]	-1.6	34	-3
32	inceptionv3	93.4 [93.2, 93.6]	95.2 [94.8, 95.6]	-1.8	30	2
33	densenet201	93.4 [93.1, 93.6]	95.2 [94.8, 95.7]	-1.9	27	6

ImageNet Top-5 TopImages						
Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
34	resnet_v1_152_tf	93.2 [92.9, 93.4]	95.2 [94.7, 95.6]	-2.0	33	1
35	resnet_v1_101_tf	92.9 [92.7, 93.1]	94.9 [94.4, 95.3]	-2.0	35	0
36	resnet50	92.9 [92.6, 93.1]	94.7 [94.2, 95.1]	-1.8	39	-3
37	resnet_v2_50_tf	92.8 [92.6, 93.1]	94.8 [94.3, 95.2]	-1.9	37	0
38	densenet169	92.8 [92.6, 93.0]	94.7 [94.2, 95.1]	-1.9	38	0
39	dpn68	92.8 [92.5, 93.0]	94.8 [94.3, 95.2]	-2.0	36	3
40	cafferesnet101	92.8 [92.5, 93.0]	94.6 [94.1, 95.0]	-1.8	40	0
41	resnet_v1_50_tf	92.2 [92.0, 92.4]	94.2 [93.8, 94.7]	-2.1	41	0
42	densenet121	92.0 [91.7, 92.2]	94.0 [93.5, 94.5]	-2.0	46	-4
43	pnasnet_mobile_tf	91.9 [91.6, 92.1]	94.1 [93.6, 94.5]	-2.2	44	-1
44	vgg19_bn	91.8 [91.6, 92.1]	94.0 [93.5, 94.4]	-2.1	47	-3
45	inception_v2_tf	91.8 [91.5, 92.0]	94.2 [93.7, 94.7]	-2.5	42	3
46	nasnetamobile	91.7 [91.5, 92.0]	94.1 [93.6, 94.5]	-2.3	43	3
47	nasnet_mobile_tf	91.6 [91.3, 91.8]	93.8 [93.4, 94.3]	-2.3	49	-2
48	bninception	91.6 [91.3, 91.8]	94.0 [93.6, 94.5]	-2.5	45	3
49	vgg16_bn	91.5 [91.3, 91.8]	93.7 [93.2, 94.1]	-2.1	50	-1
50	resnet34	91.4 [91.2, 91.7]	93.9 [93.4, 94.3]	-2.5	48	2
51	vgg19	90.9 [90.6, 91.1]	92.8 [92.2, 93.3]	-1.9	51	0
52	vgg16	90.4 [90.1, 90.6]	92.5 [92.0, 93.0]	-2.1	53	-1
53	vgg13_bn	90.4 [90.1, 90.6]	92.6 [92.1, 93.1]	-2.2	52	1
54	mobilenet_v1_tf	90.0 [89.7, 90.2]	91.4 [90.8, 91.9]	-1.4	59	-5
56	vgg_19_tf	89.8 [89.6, 90.1]	92.1 [91.5, 92.6]	-2.2	56	0
55	vgg_16_tf	89.8 [89.6, 90.1]	92.2 [91.6, 92.7]	-2.3	54	1
57	vgg11_bn	89.8 [89.5, 90.1]	91.9 [91.4, 92.5]	-2.1	58	-1
58	inception_v1_tf	89.6 [89.4, 89.9]	92.1 [91.6, 92.6]	-2.5	55	3
59	vgg13	89.2 [89.0, 89.5]	91.4 [90.8, 91.9]	-2.2	60	-1
60	resnet18	89.1 [88.8, 89.3]	92.0 [91.4, 92.5]	-2.9	57	3
61	vgg11	88.6 [88.3, 88.9]	91.0 [90.4, 91.5]	-2.4	61	0
62	squeezenet1_1	80.6 [80.3, 81.0]	83.9 [83.1, 84.6]	-3.2	62	0
63	squeezenet1_0	80.4 [80.1, 80.8]	83.5 [82.8, 84.3]	-3.1	63	0
64	alexnet	79.1 [78.7, 79.4]	81.8 [81.0, 82.6]	-2.7	64	0
65	fv_64k	55.7 [55.3, 56.2]	55.9 [54.9, 56.8]	-0.1	65	0
66	fv_16k	49.9 [49.5, 50.4]	49.8 [48.8, 50.8]	0.1	66	0
67	fv_4k	41.3 [40.8, 41.7]	41.9 [40.9, 42.8]	-0.6	67	0

3.11.0.3 Accuracy plots for all ImageNet test sets

Figure 3.12 shows the top-1 and top-5 accuracies for our three test sets and all convolutional networks in our model testbed. Figure 3.13 shows the accuracies for all models (including Fisher Vector models) with a probit scale on the axes.

3.11.0.4 Example images

Figure 3.14 shows randomly selected images for three randomly selected classes for both the original ImageNet validation set and our three new test sets.

3.11.0.5 Effect of selection frequency on model accuracy

To better understand how the selection frequency of an image impacts the model accuracies, Figures 3.15, 3.16, and 3.17 show model accuracies stratified into five selection frequency bins.

3.11.0.6 Ambiguous class examples

Figure 3.18 shows randomly selected images from the original ImageNet validation set for three pairs of classes with ambiguous class boundaries. We remark that several more classes in ImageNet have ill-defined boundaries. The three pairs of classes here were chosen only as illustrative examples.

The following list shows names and definitions for the three class pairs:

- Pair 1
    a. `projectile, missile`: "a weapon that is forcibly thrown or projected at a targets but is not self-propelled"
    b. `missile`: "a rocket carrying a warhead of conventional or nuclear explosives; may be ballistic or directed by remote control"
- Pair 2
    c. `tusker`: "any mammal with prominent tusks (especially an elephant or wild boar)"
    d. `Indian elephant, Elephas maximus`: "Asian elephant having smaller ears and tusks primarily in the male"
- Pair 3
    e. `screen, CRT screen`: "the display that is electronically created on the surface of the large end of a cathode-ray tube"

[13]Test Set A is the original validation set, Test Set B is the MatchedFrequencydataset, Test Set C is the Threshold0.7, Test set D is TopImages.

f. `monitor`: "electronic equipment that is used to check the quality or content of electronic transmissions"

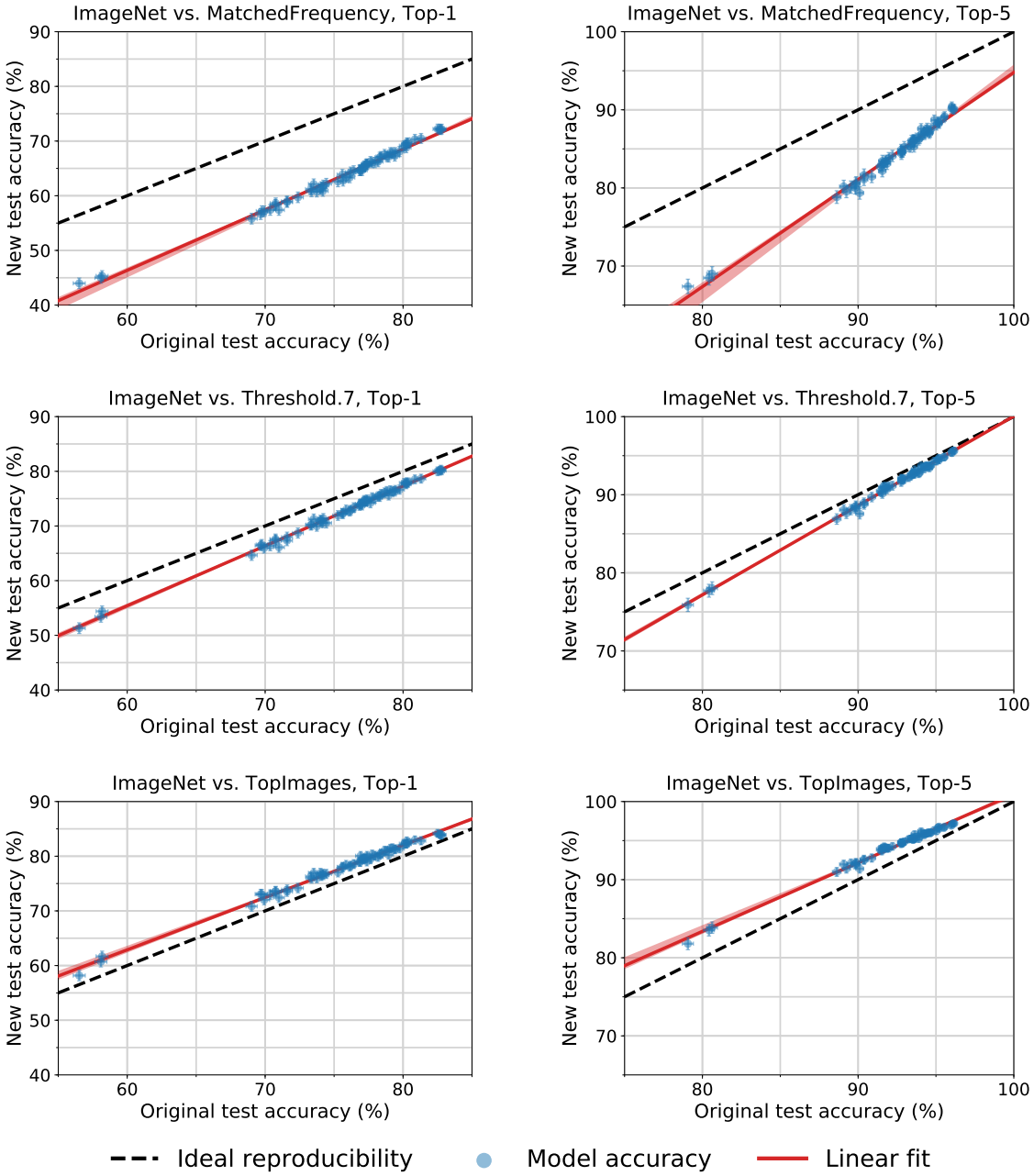


Figure 3.12: Model accuracy on the original ImageNet validation set vs. our new test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples. For MatchedFrequency, the new test set accuracies are significantly below the original accuracies. The accuracies for Threshold0.7 are still below the original counterpart, but for TopImages they improve over the original test accuracies. This shows that small variations in the data generation process can have significant impact on the accuracy scores. As for CIFAR-10, all plots reveal an approximatly linear relationship between original and new test accuracy.

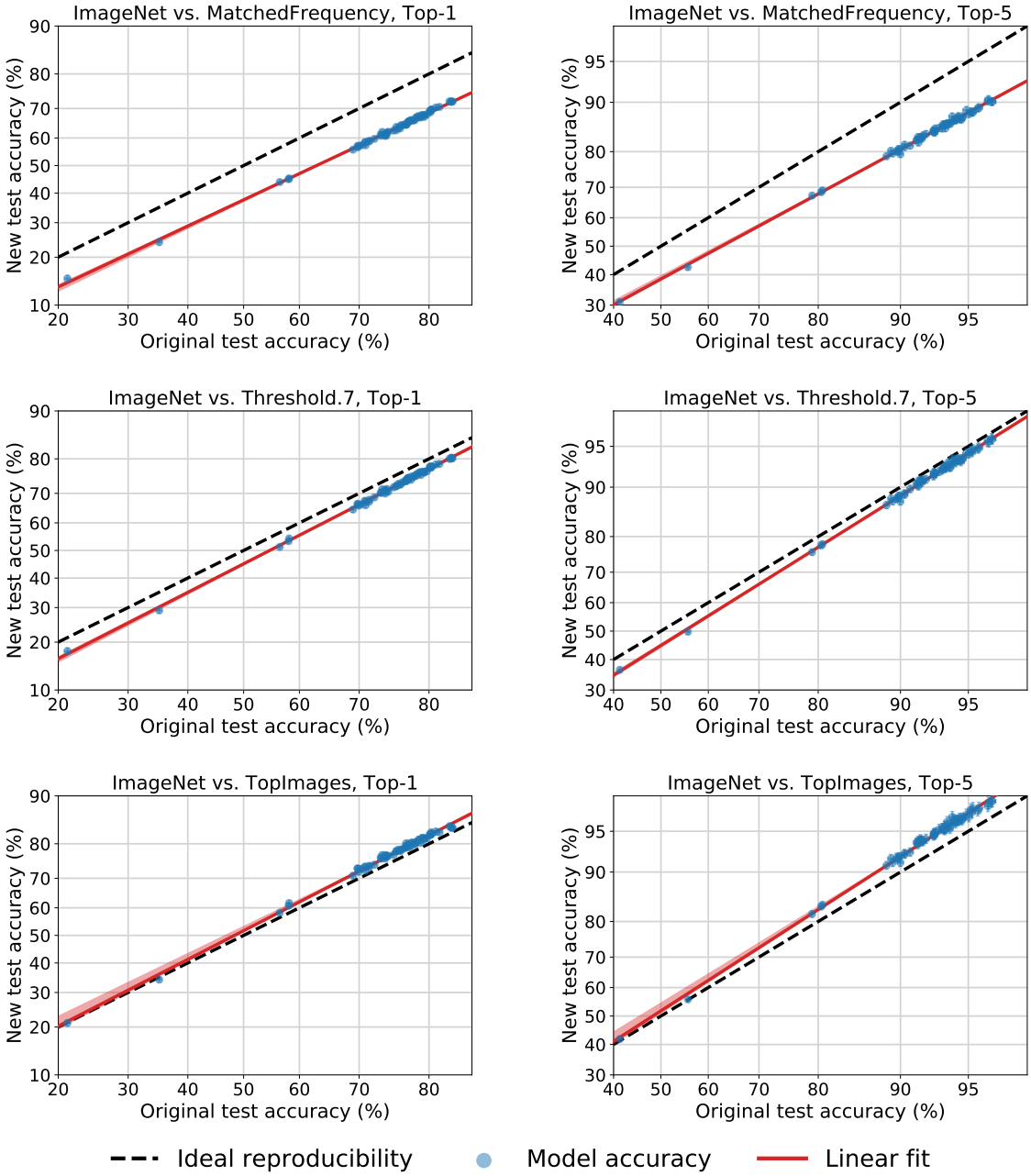


Figure 3.13: Model accuracy on the original ImageNet validation set vs. our new test sets. The structure of the plots is similar to Figure 3.12 and we refer the reader to the description there. In contrast to Figure 3.12, the plots here contain also the Fisher Vector models. Moreover, the axes are scaled according to the probit transformation, i.e., accuracy α appears at $\Phi^{-1}(\alpha)$, where Φ is the Gaussian CDF. For all three datasets and both top-1 and top-5 accuracy, the plots reveal a good linear fit in the probit domain spanning around 60 percentage points of accuracy. All plots include a 95% confidence region for the linear fit as in Figure 3.12, but the red shaded region is hard to see in some of the plots due to its small size.

(a) Test Set A

(b) Test Set B

(c) Test Set C

(d) Test Set D

Figure 3.14: Randomly selected images from the original ImageNet validation set and our new ImageNet test sets. We display four images from three randomly selected classes for each of the four datasets (the original validation set and our three test sets described in Section 3.4). The displayed classes are "Cypripedium calceolus", "gyromitra", and "mongoose". The following footnote reveals which datasets correspond to original and new ImageNet test sets.[13]

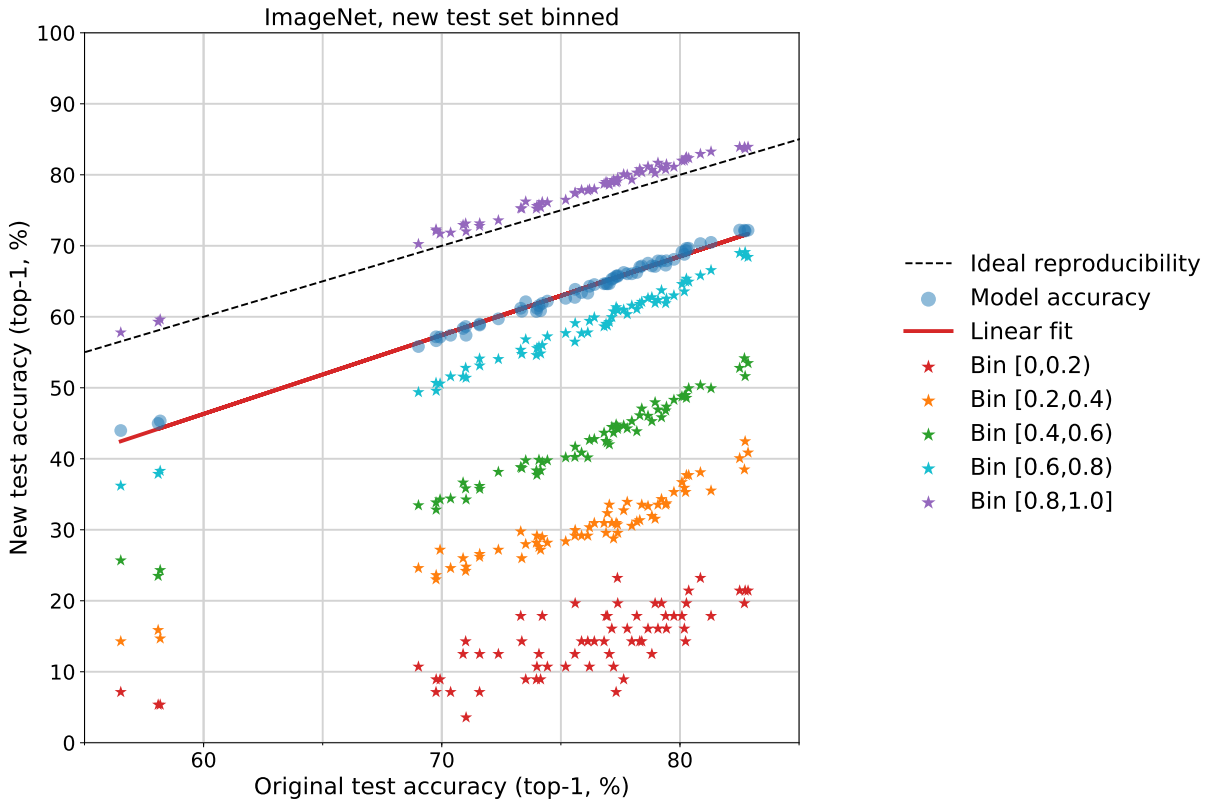


Figure 3.15: Model accuracy on our new test set MatchedFrequency, stratified into five selection frequency bins, versus the original ImageNet validation set accuracy. Every bin contains the images with MTurk selection frequency falling into the corresponding range. Each data point corresponds to one model and one of the five frequency bins (indicated by the different colors). The x-value of each data point is given by the model's accuracy on the entire original validation set. The y-value is given by the model's accuracy on our new test images falling into the respective selection frequency bin. The plot shows that the selection frequency has strong influence on the model accuracy. For instance, images with selection frequencies in the $[0.4, 0.6)$ bin lead to an average model accuracy about 20% lower than for the entire test set MatchedFrequency, and 30% lower than the original validation set. We remark that we manually reviewed all images in MatchedFrequency to ensure that (almost) all images have the correct class label, regardless of selection frequency bin.

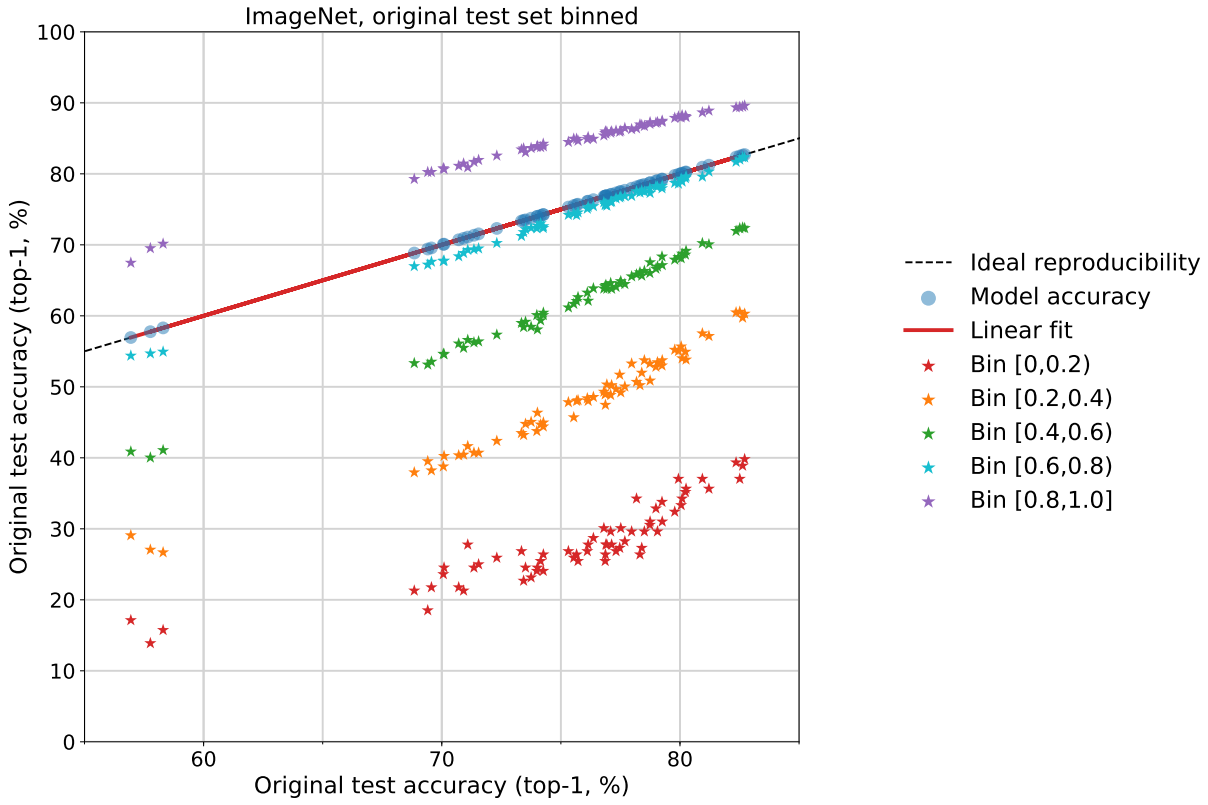


Figure 3.16: Model accuracy on the original ImageNet validation set stratified into five selection frequency bins. This plot has a similar structure as Figure 3.15 above, but contains the original validation set accuracy on both axes (as before, the images are binned on the y-axis and not binned on the x-axis, i.e., the x-value is the accuracy on the entire validation set). The plot shows that the selection frequency has strong influence on the model accuracy on the original ImageNet validation set as well. For instance, images with selection frequencies in the $[0.4, 0.6)$ bin lead to an average model accuracy about 10 – 15% lower than for the entire validation set.

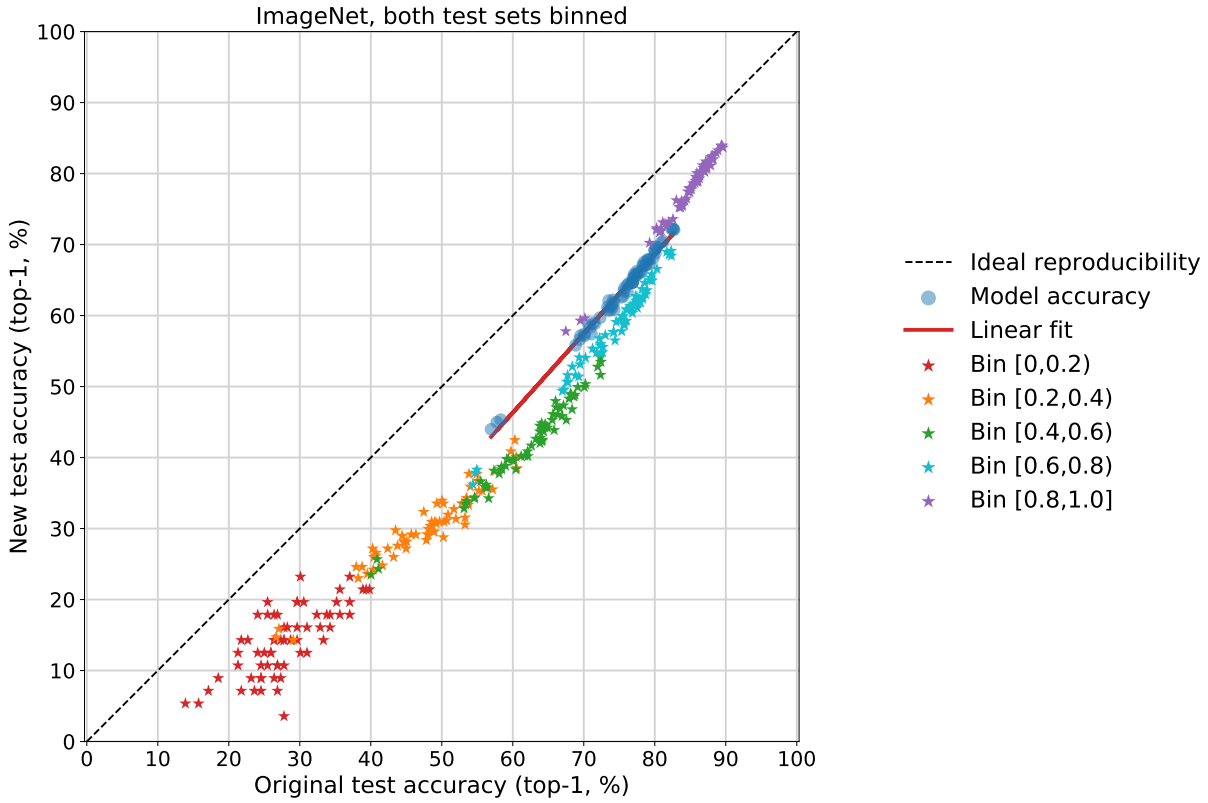
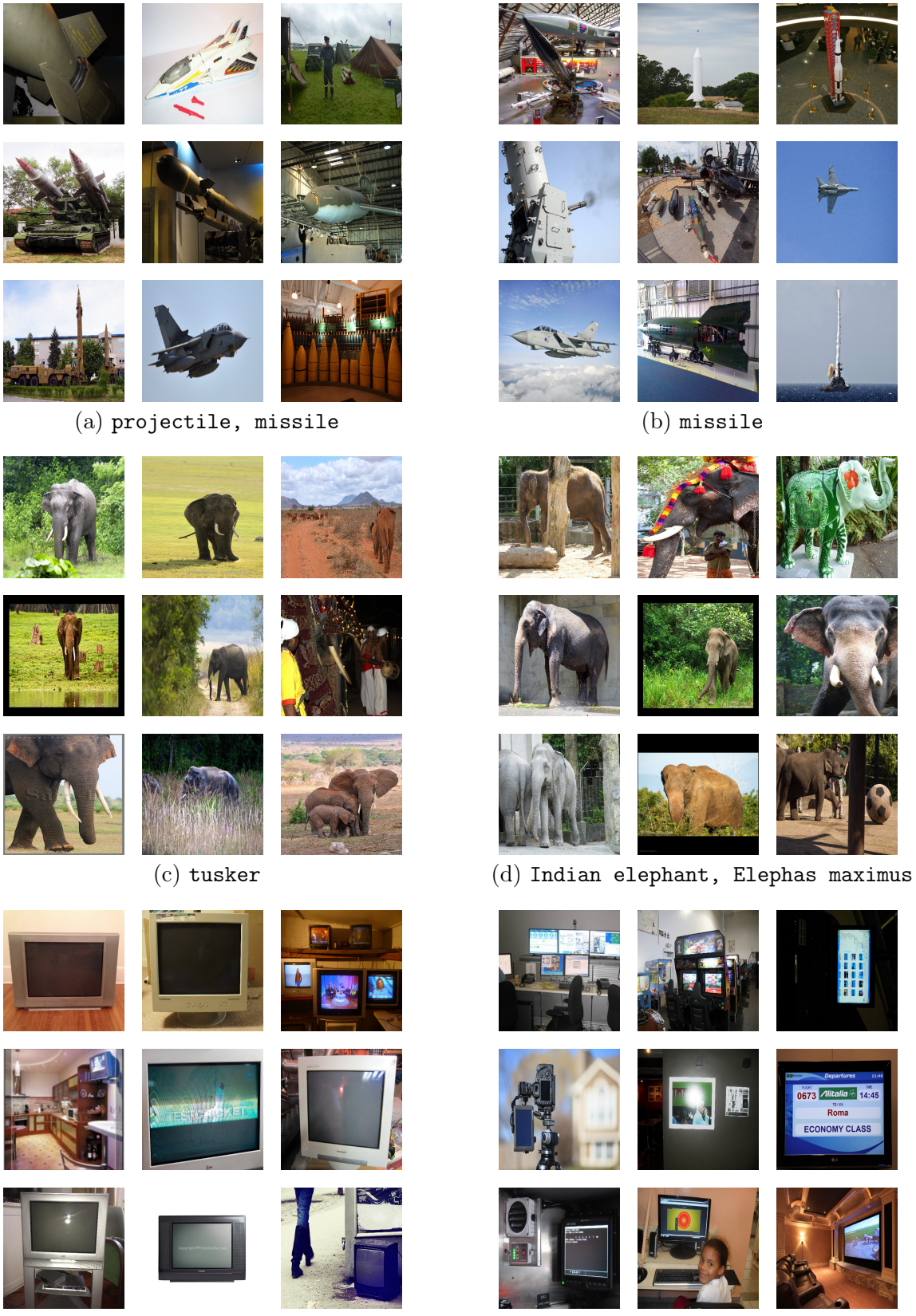


Figure 3.17: Model accuracy on the original ImageNet validation set vs. accuracy on our new test set MatchedFrequency. In contrast to the preceding Figures 3.15 and 3.16, both original and new test accuracy is now stratified into five selection frequency bins. Each data point corresponds to the accuracy achieved by one model on the images from one of the five frequency bins (indicated by the different colors). The plot shows that the model accuracies in the various bins are strongly correlated, but the accuracy on images in our new test is consistently lower. The gap is largest for images in the middle frequency bins (about 20% accuracy difference) and smallest for images in the lowest and highest frequency bins (5 – 10 % difference).



(a) `projectile, missile` (b) `missile`

(c) `tusker` (d) `Indian elephant, Elephas maximus`

(e) `screen, CRT screen` (f) `monitor`

Figure 3.18: Random images from the original ImageNet validation set for three pairs of classes with ambiguous class boundaries.

Chapter 4

A meta-analysis of overfitting in machine learning

4.1 Introduction

The holdout method is central to empirical progress in the machine learning community. Competitions, benchmarks, and large-scale hyperparameter search all rely on splitting a data set into multiple pieces to separate model training from evaluation. However, when practitioners repeatedly reuse holdout data, the danger of overfitting to the holdout data arises [67, 20].

Despite its importance, there is little empirical research into the manifested robustness and validity of the holdout method in practical scenarios. Real-world use cases of the holdout method often fall outside the guarantees of existing theoretical bounds, making questions of validity a matter of guesswork.

Recent replication studies [78] demonstrated that the popular CIFAR-10 [53] and ImageNet [15, 81] benchmarks continue to support progress despite years of intensive use. The longevity of these benchmarks perhaps suggests that overfitting to holdout data is less of a concern than reasoning from first principles might have suggested. However, this is evidence from only two, albeit important, computer vision benchmarks. It remains unclear whether the observed phenomenon is specific to the data domain, model class, or practices of vision researchers. Unfortunately, these replication studies required assembling new test sets from scratch, resulting in a highly labor-intensive analysis that is difficult to scale.

In this thesis, we empirically study holdout reuse at a significantly larger scale by analyzing data from 112 machine learning competitions on the popular Kaggle platform [44]. Kaggle competitions are a particularly well-suited environment for studying overfitting since data sources are diverse, contestants use a wide range of model families, and training techniques vary greatly. Moreover, Kaggle competitions use public and private test data splits which provide a natural experimental setup for measuring overfitting on various datasets.

To provide a detailed analysis of each competition, we introduce a coherent methodology

to characterize the extent of overfitting at three increasingly fine scales. Our approach allows us both to discuss the overall "health" of a competition across all submissions and to inspect signs of overfitting separately among the top submissions. In addition, we develop a statistical test specific to the classification competitions on Kaggle to compare the submission scores to those arising in an ideal null model that assumes no overfitting. Observed data that are close to data predicted by the null model is strong evidence against overfitting.

Overall, we conclude that the classification competitions on Kaggle show little to no signs of overfitting. While there are some outlier competitions in the data, these competitions usually have pathologies such as non-i.i.d. data splits or (effectively) small test sets. Among the remaining competitions, the public and private test scores show a remarkably good correspondence. The picture becomes more nuanced among the highest scoring submissions, but the overall effect sizes of (potential) overfitting are typically small (e.g., less than 1% classification accuracy). Thus, our findings show that substantial overfitting is unlikely to occur naturally in regular machine learning workflows.

4.2 Background and setup

Before we delve into the analysis of the Kaggle data, we briefly define the type of overfitting we study and then describe how the Kaggle competition format naturally lends itself to investigating overfitting in machine learning competitions.

4.2.1 Adaptive overfitting

"Overfitting" is often used as an umbrella term to describe any unwanted performance drop of a machine learning model. Here, we focus on *adaptive* overfitting, which is overfitting caused by test set reuse. While other phenomena under the overfitting umbrella are also important aspects of reliable machine learning (e.g. performance drops due to distribution shifts), they are beyond the scope of our paper since they require an experimental setup different from ours.

Formally, let $f : \mathcal{X} \to \mathcal{Y}$ be a trained model that maps examples $x \in \mathcal{X}$ to output values $y \in \mathcal{Y}$ (e.g., class labels or regression targets). The standard approach to measuring the performance of such a trained model is to define a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and to draw samples $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ from a data distribution $\mathcal{D}$ which we then use to evaluate the test loss $L_S(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$. As long as the model f does not depend on the test set S, standard concentration results [83] show that $L_S(f)$ is a good approximation of the true performance given by the population loss $L_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}[l(f(x), y)]$.

However, machine learning practitioners often undermine the assumption that f does not depend on the test set by selecting models and tuning hyperparameters based on the test loss. Especially when algorithm designers evaluate a large number of different models on the same test set, the final classifier may only perform well on the specific examples in the

test set. The failure to generalize to the entire data distribution $\mathcal{D}$ manifests itself in a large *adaptivity gap* $L_{\mathcal{D}}(f) - L_S(f)$ and leads to overly optimistic performance estimates.

4.2.2 Kaggle

Kaggle is the most widely used platform for machine learning competitions, currently hosting 1,461 active and completed competitions. Various organizations (companies, educators, etc.) provide the datasets and evaluation rules for the competitions, which are generally open to any participant. Each competition is centered around a dataset consisting of a training set and a test set.

Considering the danger of overfitting to the test set in a competitive environment, Kaggle subdivides each test set into public and private components. The subsets are randomly shuffled together and the entire test set is released without labels, so that participants should not know which test samples belong to which split. Hence participants submit predictions for the *entire* test set. The Kaggle server then internally evaluates each submission on both public and private splits and updates the public competition leaderboard only with the score on the public split. At the end of the competition, Kaggle releases the private scores, which determine the winner.

Kaggle has released the MetaKaggle dataset[1], which contains detailed information about competitions, submissions, etc. on the Kaggle platform. The structure of Kaggle competitions makes MetaKaggle a useful dataset for investigating overfitting empirically at a large scale. In particular, we can view the public test split S_{public} as the regular test set and use the held-out private test split S_{private} to approximate the population loss. Since Kaggle competitors do not receive feedback from S_{private} until the competition has ended, we assume that the submitted models may be overfit to S_{public} but not to S_{private}.[2] Under this assumption, the difference between private and public loss $L_{S_{\text{private}}}(f) - L_{S_{\text{public}}}(f)$ is an approximation of the adaptivity gap $L_{\mathcal{D}}(f) - L_S(f)$. Hence our setup allows us to estimate the amount of overfitting occurring in a typical machine learning competition. In the rest of this paper, we will analyze the public versus private score differences as a proxy for adaptive overfitting.

Due to the large number of competitions on the Kaggle platform, we restrict our attention to the most popular classification competitions. In particular, we survey the competitions with at least 1,000 submissions before the competition deadline. Moreover, we include only competitions with evaluation metrics that have at least 10 competitions. These metrics are classification accuracy, AUC (area under curve), MAP@K (mean average precision), the logistic loss, and a multiclass variant of the logistic loss. Section 4.3 provides more detail on our selection criteria.

[1] `https://www.kaggle.com/kaggle/meta-kaggle`

[2] Since test examples without labels are available, contestants may still overfit to S_{private} using an unsupervised approach. While such overfitting may have occurred in a limited number of competitions, we base our analysis on the assumption that unsupervised overfitting did not occur widely with effect sizes that are large compared to overfitting to the public test split.

4.3 Overview of Kaggle competitions and our resulting selection criteria

At the time of writing, the MetaKaggle dataset has 1,461 competitions. Due to the variety of different scoring mechanisms used on Kaggle and intricacies of individual competitions, it is challenging to analyze the entire datast in a coherent way. To limit the scope of our investigation and allow for a detailed analysis, we restricted our attention to a subset of competitions based on the following two criteria.

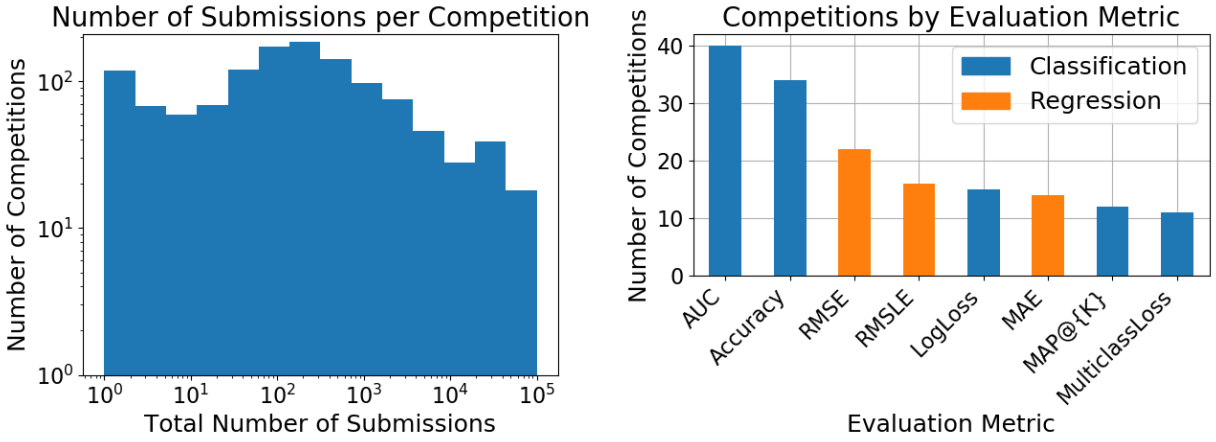


Figure 4.1: Overview of the Kaggle competitions. The left plot shows the distribution of submissions per competition. The right plot shows the score types that are most common among the competitions with at least 1,000 submissions.

Number of submissions. Kaggle competitions have a wide range of submissions, ranging from 0 for the bottom 10% to 3,159 for the most popular 10% of competitions (and 132,097 for the most popular competition). The left plot in Figure 4.1 shows the distribution of competition sizes. From the perspective of adaptive overfitting, competitions with many submissions are more interesting since a larger number of submissions indicate more potential adaptivity. Moreover, overfitting on the most popular competitions would also be more important to understand as it would undermine the credibility of the competition paradigm. So to limit the number of competitions, we included only competitions with at least 1,000 submissions before the corresponding competition deadline. This left a total of 262 competitions.

Score type. Depending on the competition, the performance of a submissions is measured with one of various scoring mechanisms. The list of scoring mechanisms involves well-known score types (accuracy, AUC, RMSE) and a long tail of more specialized metrics (see the right

plot in Figure 4.1. We focus on score types used for classification tasks with at least 10 competitions (passing our submission count filter) so we can compare competitions with multiple other competitions using the same score type. This avoids difficulties such as comparing classification vs. regression competitions or bounded vs. unbounded scores. Moreover, the most popular score types are also the most relevant for measuring the amount of overfitting in machine learning competitions. The classification score types with at least 10 competitions are classification accuracy, AUC (area under curve), MAP@K (mean average precision), the logistic loss, and a multiclass variant of the logistic loss.

In total, the above exclusion rules leave 112 competitions. It would be possible to restrict the set of competitions even further, e.g., by only including "featured" competitions (the competitions promoted by Kaggle, usually with prize money) and excluding other competition types such as "in class" competitions which are mainly used for educational purposes. Another possible filter would be to exclude competitions with only small test set sizes since the resulting scores are inherently more noisy. However, we decided to minimize the exclusion rules so we can analyze a large number of competitions and possibly find effects of features such as competition type and test set size.

4.4 Detailed analysis of competitions scored with classification accuracy

We begin with a detailed look at classification competitions scored with the standard accuracy metric. Classification is the prototypical machine learning task and accuracy is a widely understood performance measure. This makes the corresponding competitions a natural starting point to understand nuances in the Kaggle data. Moreover, there is a large number of accuracy competitions, which enables us to meaningfully compare effect sizes across competitions. As we will see in Section 4.4.3, accuracy competitions also offer the advantage that we can obtain measures of statistical uncertainty. Later sections will then present an overview of the competitions scored with other metrics.

We conduct our analysis of overfitting at three levels of granularity that become increasingly stringent. The first level considers all submissions in a competition and checks for systematic overfitting that would affect a substantial number of submissions (e.g., if public and private score diverge early in the competition). The second level then zooms into the top 10% of submissions (measured by public accuracy) and conducts a similar comparison of public to private scores. The goal here is to understand whether there is more overfitting among the best submissions since they are likely most adapted to the test set. The third analysis level then takes a mainly quantitative approach and computes the probabilities of the observed public vs. private accuracy differences under an ideal null model. This allows us to check if the observed gaps are larger than purely random fluctuations.

In the following subsections, we will apply these three analysis methods to investigate four accuracy competitions. These four competitions are the accuracy competitions with

the largest number of submissions and serve as representative examples for a typical accuracy competition in the MetaKaggle dataset (see Table 4.1 for information about these competitions). Section 4.4.5 then complements these analyses with a quantitative look at all competitions before we summarize our findings for accuracy competitions in 4.4.6.

Table 4.1: The four Kaggle accuracy competitions with the largest number of submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set.

ID	Name	# Submissions	n_{public}	n_{private}
5275	Can we predict voting outcomes?	35,247	249,344	249,343
3788	Allstate Purchase Prediction Challenge	24,532	59,657	139,199
7634	TensorFlow Speech Recognition Challenge	24,263	3,171	155,365
7115	Cdiscount's Image Classification Challenge	5,859	53,0455	1,237,727

4.4.1 First analysis level: visualizing the overall trend

As mentioned in Section 4.2.2, the main quantities of interest are the accuracies on the public and private parts of the test set. In order to visualize this information at the level of all submissions to a competition, we create a scatter plot of the public and private accuracies. In an ideal competition with a large test set and without any overfitting, the public and private accuracies of a submission would all be almost identical and lie near the $y = x$ diagonal. On the other hand, substantial deviations from the diagonal would indicate gaps between the public and private accuracy and present possible evidence of overfitting in the competition.

Figure 4.2 shows such scatter plots for the four accuracy competitions mentioned above. In addition to a point for each submission, the scatter plots also contain a linear regression fit to the data. All four plots show a linear fit that is close to the $y = x$ diagonal. In addition, three of the four plots show very little variation around the diagonal. The variation around the diagonal in the remaining competition is largely symmetric, which indicates that it is likely the effect of random chance.

These scatter plots can be seen as indicators of overall competition "health": in case of pervasive overfitting, we would expect a plateauing trend where later points mainly move on the x-axis (public accuracy) but stagnate on the y-axis (private accuracy). In contrast, the four plots in Figure 4.2 show that as submissions progress on the public test set, they see corresponding improvements also on the private test set. Moreover, the public scores remain representative of the private scores.

As can be seen in Section 4.8.1.3, this overall trend is representative for the 34 accuracy competitions. All except one competition show a linear fit close to the main diagonal. The only competition with a substantial deviation is the "TAU Robot Surface Detection" competition (ID 12598). We contacted the authors of this competition and confirmed that there are subtleties in the public / private split which undermine the assumption that the

two splits are i.i.d. Hence we consider this competition to be an outlier since it does not conform to the experimental setup described in Section 4.2.2. So at least on the coarse scale of the first analysis level, there are little to no signs of adaptive overfitting: it is easier to make genuine progress on the data distribution in these competitions than to substantially overfit to the test set.

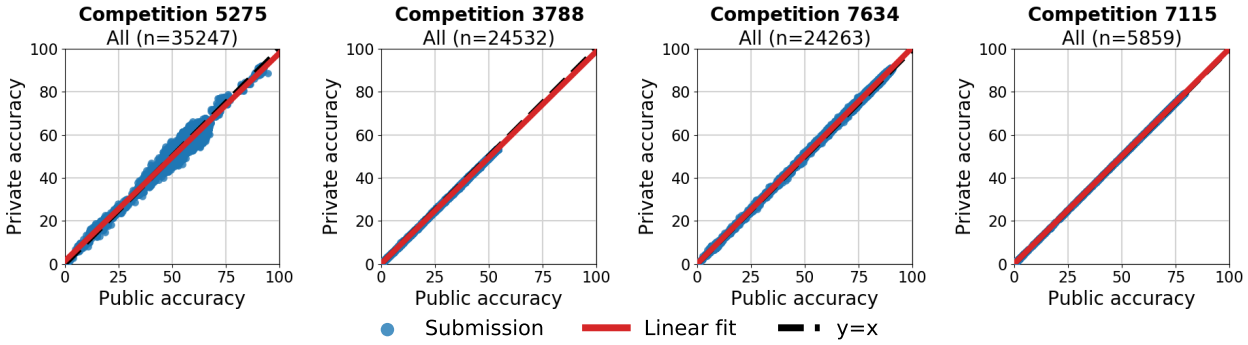


Figure 4.2: Private versus public accuracy for all submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

4.4.2 Second analysis level: zooming in to the top submissions

While the scatter plots discussed above give a comprehensive picture of an entire competition, one concern is that overfitting may be more prevalent among the submissions with the highest public accuracy since they may be more adapted to the public test set. Moreover, the best submissions are those where overfitting would be most serious since invalid accuracies there would give a misleading impression of performance on future data. So to analyze the best submissions in more detail, we also created scatter plots for the top 10% of submissions (as scored by public accuracy).

Figure 4.3 shows scatter plots for the same four competitions as before. Since the axes now encompass a much smaller range (often only a few percent), they give a more nuanced picture of the performance among the best submissions.

In the leftmost and rightmost plot, the linear fit for the submissions still closely tracks the $y = x$ diagonal. Hence there is little sign of overfitting also among the top 10% of submissions. On the other hand, the middle two plots in Figure 4.3 show noticeable deviations from the main diagonal. Interestingly, the linear fit is *above* the diagonal in Competition 7634 and *below* the main diagonal in Competition 3788. The trend in Competition 3788 is more concerning since it indicates that the public accuracy overestimates the private accuracy. However, in both competitions the absolute effect size (deviation from the diagonal) is small (about 1%). It is also worth noting that the accuracies in Competition 3788 are not in a high-accuracy regime but around 55%. So the relative error from public to private test set is small as well.

Section 4.8.1.4 contains scatter plots for the remaining accuracy competitions that show a similar overall trend. Besides the Competition 12598 discussed in the previous subsection, there are two additional competitions with a substantial public vs. private deviation. One competition is #3641, which has a total test set size of about 4,000 but is only derived from 7 human test subjects (the dataset consists of magnetoencephalography (MEG) recordings from these subjects). The other competition is 12681, which contains a public test set of size 90. Since very small (effective) test sets make it easier to reconstruct the public / private split and then overfit, and make the public and private scores noisier, we consider these two competitions to be outliers. Overall, the second analysis level shows that if overfitting occurs among the top submissions, it only does so to a small extent.

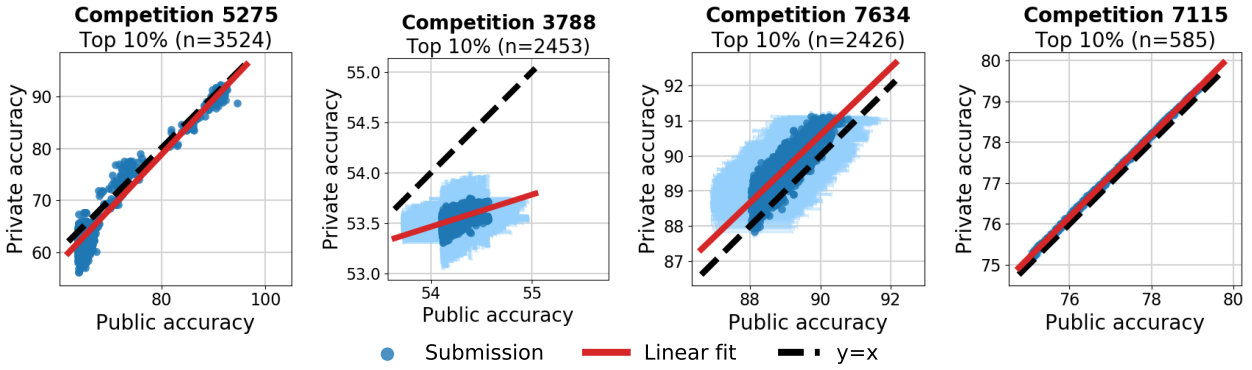


Figure 4.3: Private versus public accuracy for the top 10% of submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

4.4.3 Third analysis level: quantifying the amount of random variation

When discussing deviations from the ideal $y = x$ diagonal in the previous two subsections, an important question is how much variation we should expect from random chance. Due to the finite sizes of the test sets, the public and private accuracies will never match exactly and it is a priori unclear how much of the deviation we can attribute to random chance and how much to overfitting.

To quantitatively understand the expected random variation, we compute the probability of a given public vs. private deviation (*p-value*) for a simple null model. By inspecting the distribution of the resulting p-values, we can then investigate to what extent the observed deviations can be attributed to random chance.

We consider the following null model under which we compute p-values for observing certain gaps between the public and private accuracies. We fix a submission that makes a given number of mistakes on the *entire* test set (public and private split combined). We then randomly split the test set into two parts with sizes corresponding to the public and private

splits of the competition. This leads to a certain number of mistakes (and hence accuracy) on each of the two parts. The p-value for this submission is then given by the probability of the event

$$|\text{public\_accuracy} - \text{private\_accuracy}| \geq \varepsilon \tag{4.4.1}$$

where ε is the observed deviation between public and private accuracy. We describe the details of computing these p-values in Section .

Figure 4.4 plots the distribution of these p-values for the same four competitions as before. To see potential effects of overfitting, we show p-values for all submissions, the top 10% of submissions (as in the previous subsection), and for the first submission for each team in the competition. We plot the first submissions separately since they should be least adapted to the test set and hence show the smallest amount of overfitting.

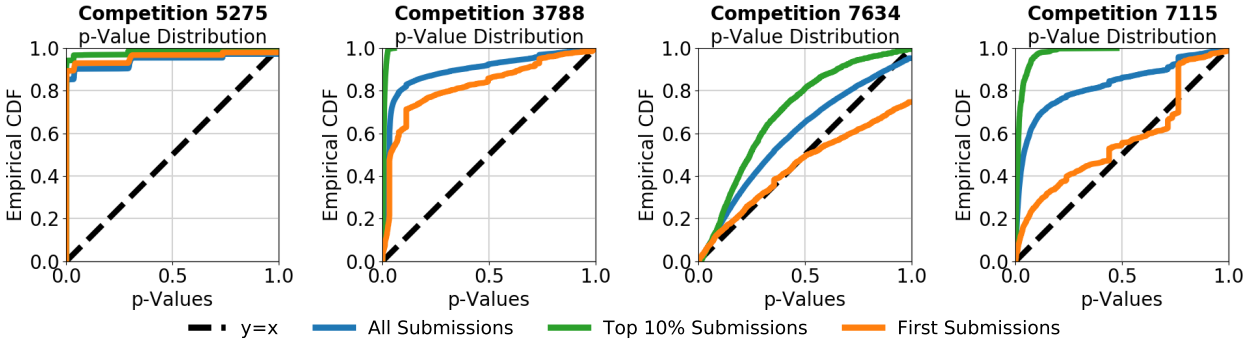


Figure 4.4: Empirical CDFs of the p-values for three (sub)sets of submissions in the four accuracy competitions with the largest number of submissions.

Under our ideal null hypothesis, the p-values would have a uniform distribution with a CDF following the $y = x$ diagonal. However, this is only the case for the first submissions in Competitions 7634 and 7115, and even there only approximately. Most other curves show a substantially larger number of small p-values (large gaps between public and private accuracy) than expected under the null model. Moreover, the top 10% of submissions always exhibit the smallest p-values, followed by all submissions and then the first submissions.

When interpreting these p-value plots, we emphasize that the null model is highly idealized. In particular, we assume that every submission is evaluated on its own *independent* random public / private split, which is clearly not the case for Kaggle competitions. So for correlated submissions, we would expect clusters of approximately equal p-values that are unlikely to arise under the null model. Since it is plausible that many models are trained with standard libraries such as XGBoost [5] or scikit-learn [70], we conjecture that correlated models are behind the jumps in the p-value CDFs.

In addition, it is important to note that for large test sets (e.g., the 498,000 examples in Competition 5275), even very small systematic deviations between the public and private accuracies are statistically significant and hence lead to small p-values. So while the analysis based on p-value plots does point towards irregularities such as overfitting, the overall effect size (deviation between public and private accuracies) can still be small.

Given the highly discriminative nature of the p-values, a natural question is whether any competition exhibits approximately uniform p-values. As can be seen in Section 4.8.1.5, some competitions indeed have p-value plots that are close to the uniform distribution under null model (Figure 4.10 in the same section highlights four examples). Due to the idealized null hypothesis, this is strong evidence that these competitions are free from overfitting.

4.4.4 Computation of p-values

For a given submission, the public and private test accuracies will rarely exactly match. We would like to understand exactly how much of this discrepancy can be attributed to random variation as opposed to overfitting. One approach to this problem is to estimate the model's accuracy on the public test-set and then plug the observed test-set accuracy into a binomial tail bound. However, this method is not fully rigorous. For instance, how do we account for uncertainty in the model's test-accuracy estimate? Fortunately, we can side-step these issues and compute *exact* p-values for a simple null model using only MetaKaggle data.

Fix a classifier that correctly classifies m points out of a total test set of size n_{total} and incorrectly classifies the remaining $n_{\text{total}} - m$ points. We randomly split these n_{total} points into public and private test sets of size n_{public} and n_{private}, respectively, with $n_{\text{public}} + n_{\text{private}} = n_{\text{total}}$. Under the null model, every partition of the points is equally likely. Dividing points in this fashion leads to m_{public} and m_{private} successes on each subset. Let Δ be the observed difference in public and private accuracies. Then, we're interested in computing

$$\text{p\_value} = \mathbb{P}\left\{\left|\frac{m_{\text{private}}}{n_{\text{private}}} - \frac{m_{\text{public}}}{n_{\text{public}}}\right| > \Delta\right\}. \tag{4.4.2}$$

Fixing m_{private} also determines m_{public}, and for any fixed $m_{\text{private}} = k$, the probability of sampling a private test set of with k correctly classified points is

$$\frac{\binom{m}{k}\binom{n-m}{n_{\text{private}}-k}}{\binom{n}{n_{\text{private}}}}. \tag{4.4.3}$$

Therefore, we can compute the p-value in equation (4.4.2) as

$$\mathbb{P}\left\{\left|\frac{m_{\text{private}}}{n_{\text{private}}} - \frac{m_{\text{public}}}{n_{\text{public}}}\right| > \Delta\right\} = \sum_{k\in[m]:\left|\frac{k}{n_{\text{private}}} - \frac{(m-k)}{n_{\text{public}}}\right| > \Delta} \frac{\binom{m}{k}\binom{n-m}{n_{\text{private}}-k}}{\binom{n}{n_{\text{private}}}}, \tag{4.4.4}$$

where $[m] = \{0, 1, 2, \ldots, m\}$. Evaluating this summation only requires access to $n_{\text{total}}, n_{\text{public}}, n_{\text{private}}$, and Δ, all of which are available in the MetaKaggle dataset. Hence, a p-value can be readily evaluated for each submission. Computing ratios of binomial coefficients like (4.4.3) is numerically unstable and computationally expensive for large n. Therefore, in our implementation, we work in log-space and use Stirling's approximation for the binomial coefficients to efficiently and reliably approximate (4.4.4).

4.4.5 Aggregate view of the accuracy competitions

The previous subsections provided tools for analyzing individual competitions and then relied on a qualitative survey of all accuracy competitions. Due to the many nuances and failure modes in machine learning competitions, we believe that this case-by-case approach is the most suitable for the Kaggle data. But since this approach also carries the risk of missing overall trends in the data, we complement it here with a quantitative analysis of all accuracy competitions.

In order to compare the amount of overfitting across competitions, we compute the *mean accuracy difference* across all submissions in a competition. Specifically, let $\mathcal{C}$ be a set of submissions for a given competition. Then the mean accuracy difference of the competition is defined as

$$\text{mean\_accuracy\_difference} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \text{public\_accuracy}(i) - \text{private\_accuracy}(i) \; . \quad (4.4.5)$$

A larger mean accuracy difference indicates more potential overfitting. Note that since accuracy is in a sense the opposite of loss, our computation of mean accuracy difference is the opposite of our earlier expression for the generalization gap.

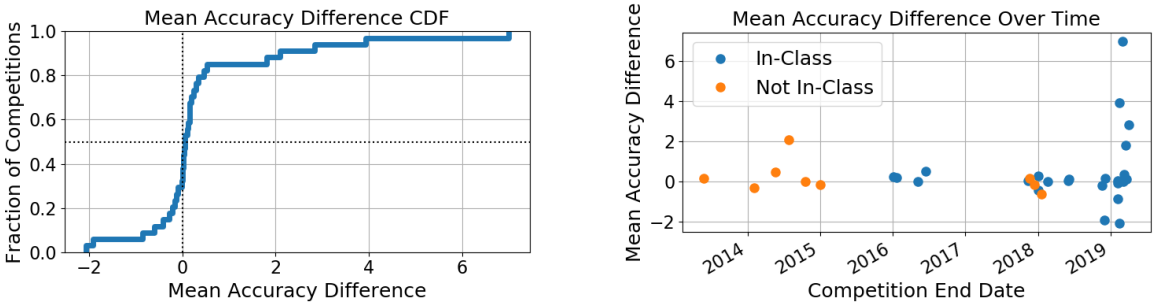


Figure 4.5: Left: Empirical CDF of the mean accuracy differences (%) for 34 accuracy competitions. Right: Mean accuracy differences versus competition end date for the same competitions.

Figure 4.5 shows the empirical CDF of the mean accuracy differences for all accuracy competitions. While the score differences between -2% to $+2\%$ are approximately symmetric and centered at zero (as a central limit theorem argument would suggest), the plot also shows a tail with larger score differences consisting of five competitions. Figure 4.7 in Section 4.8.1.2 aggregates our three types of analysis plots for these competitions.

We have already mentioned three of these outliers in the discussion so far. The worst outlier is Competition 12598 which contains a non-i.i.d. data split (see Section 4.4.1). Section 4.4.2 noted that Competitions 3641 and 12681 had very small test sets. Similarly, the other two outliers (#12349 and #5903) have small private test sets of size 209 and 100 respectively. Moreover, the public - private accuracy gap *decreases* among the very best submissions in

these competitions (see Figure 4.7 in Section 4.8.1.2). Hence we do not consider these competitions as examples of adaptive overfitting.

As a second aggregate comparison, the right plot in Figure 4.5 shows the mean accuracy differences vs. competition end date and separates two types of competitions: in-class and other, which are mainly the "featured" competitions on Kaggle with prize money. Interestingly, many of the competitions with large public - private deviations are from 2019. Moreover, the large deviations are almost exclusively in-class competitions. This may indicate that in-class competitions undergo less quality control than the featured competitions (e.g., have smaller test sets), and that the quality control standards on Kaggle may change over time.

4.4.6 Did we observe overfitting?

The preceding subsections provided an increasingly fine-grained analysis of the Kaggle competitions evaluated with classification accuracy. The scatter plots for all submissions in Section 4.4.1 show a good fit to the $y = x$ diagonal with small and approximately symmetric deviations from the diagonal. This is strong evidence that the overall competition is not affected by substantial overfitting. When restricting the plots to the top submissions in Section 4.4.2, the picture becomes more varied but the largest overfitting effect size (public - private accuracy deviation) is still small. In both sections we have identified outlier competitions that do not follow the overall trend but also have issues such as small test sets or non-i.i.d. splits. At the finest level of our analysis (Section 4.4.3), the p-value plots show that the data is only sometimes in agreement with an idealized null model for no overfitting.

Overall we see more signs of overfitting as we sharpen our analysis to highlight smaller effect sizes. So while we cannot rule out every form of overfitting, we view our findings as evidence that overfitting did not pose a significant danger in the most popular classification accuracy competitions on Kaggle. In spite of up to 35,000 submissions to these competitions, there are no large overfitting effects. So while overfitting may have occurred to a small extent, it did not invalidate the overall conclusions from the competitions such as which submissions rank among the top or how well they perform on the private test split.

4.5 Classification competitions with further evaluation metrics

In addition to accuracy classification competitions, we also surveyed competitions evaluated with AUC, MAP@K, LogLoss, and MulticlassLoss. Unfortunately, the Meta Kaggle dataset contains only *aggregate* scores for the public and private test set splits, not the loss values for individual predictions. For the accuracy metric, these quantities are sufficient to compute statistical measures of uncertainty such as the error bars in the scatter plots (Sections 4.4.1 & 4.4.2) or the p-value plots (Section 4.4.3). However, the aggregate data is not enough to

compute similar quantities for the other classification metrics. For instance, lack of example-level scores precludes the use of standard tools such as the bootstrap or permutation tests. Hence our analysis here is more qualitative than for accuracy competitions in the preceding section. Nevertheless, inspecting the scatter plots can still convey overall trends in the data.

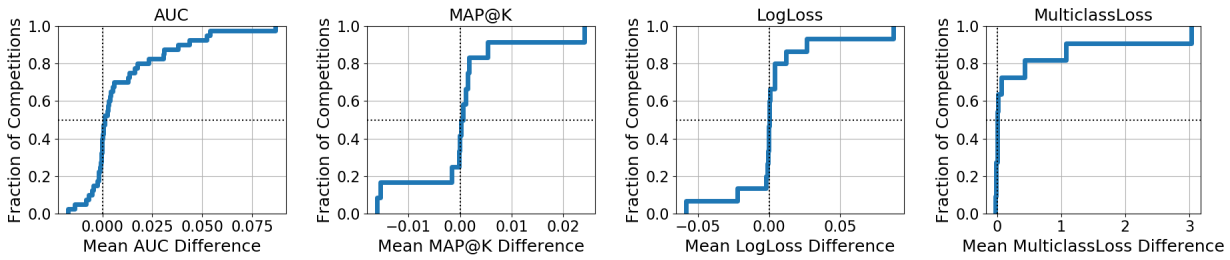


Figure 4.6: Empirical CDF of mean score differences for 40 AUC competitions, 12 MAP@K competitions, 15 LogLoss competitions, and 11 MulticlassLoss competitions.

Sections 4.8.2 to 4.8.5 contain the scatter plots for all submissions and for the top 10% of submissions to these competitions. The overall picture is similar to the accuracy plots: many competitions have scatter plots with a good linear fit close to the $y = x$ diagonal. There is more variation in the top 10% plots but due to the lack of error bars it is difficult to attribute this to overfitting vs. random noise. As before, a small number of competitions show more variation that may be indicative of overfitting. In all cases, the Kaggle website (data descriptions and discussion forums) give possible reasons for this behavior (non-i.i.d. splits, competitions with two stages and different test sets, etc.). Thus, we view these competitions as outliers that do not contradict the overall trend.

Figure 4.6 shows plots with aggregate statistics (mean score difference) similar to Section 4.4.5. As for the accuracy competitions, the empirical distribution has an approximately symmetric part centered at 0 (no score change) and a tail with larger score differences that consists mainly of outlier competitions.

4.6 Related work

As mentioned in the introduction, the reproducibility experiment of Recht et. al. [78] also points towards a surprising absence of adaptive overfitting in popular machine learning benchmarks. However, there are two important differences to our work. First, Recht et. al. [78] assembled new test sets from scratch, which makes it hard to disentangle the effects of adaptive overfitting and distribution shifts. In contrast, most of the public / private splits in Kaggle competitions are i.i.d., which removes distribution shifts as a confounder. Second, they only investigate two image classification benchmarks on which most models come from the same model class (CNNs) [27, 54]. We survey 112 competitions on which the Kaggle competitors experimented with a broad range of models and training approaches. Hence our conclusions about overfitting apply to machine learning more broadly.

The adaptive data analysis literature [20, 80] provides a range of theoretical explanations for how the common machine learning workflow may implicitly mitigate overfitting [4, 105, 26]. Our work is complementary to these papers and conducts a purely empirical study of overfitting in machine learning competitions. We hope that our findings can help test and refine the theoretical understanding of overfitting in future work.

4.7 Conclusion and future work

We surveyed 112 competitions on Kaggle covering a wide range of classification tasks but found little to no signs of overfitting. Our results cast doubt on the standard narrative that overfitting is a significant danger in the common machine learning workflow.

Moreover, our findings call into question whether common practices such as limiting test set re-use increase the reliability of machine learning. We have seen multiple competitions where a non-i.i.d. split lead to substantial gaps between public and private scores, suggesting that distribution shifts [73, 92, 78, 22] may be a more pressing problem than test set reuse in current machine learning.

4.8 Supplementary material

4.8.1 Accuracy

4.8.1.1 Accuracy: competition info

Table 4.2: Competitions scored with accuracy with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes. A * after the competition name means the name was slightly edited to fit into the table.

Accuracy				
ID	Name	# Sub.	n_{public}	n_{private}
3362	Dogs vs. Cats	1,225	3,750	8,751
3366	The Black Box Learning Challenge*	1,924	5,000	5,000
3428	Data Science London + Scikit-learn	1,477	2,700	6,299
3641	DecMeg2014 - Decoding the Human Brain	4,507	1,745	2,313
3649	CIFAR-10 - Object Recognition in Images	1,634	9,000	291,000
3788	Allstate Purchase Prediction Challenge	24,532	59,657	139,199
4554	MSU Visits*	1,288	30,000	270,000
4821	104-1 MLDS_Final	1,060	36,400	36,401
4857	Let's Overfit	1,080	500,000	500,000
5275	Can we predict voting outcomes?	35,247	249,344	249,343
5896	Prediction Reviews Sentiment Analysis*	1,770	50	50
5903	Prediction Reviews Sentiment Analysis (Light)*	3,528	400	100
5942	Identify Me If You Can – Yandex & MIPT	1,242	20,588	20,589
6109	Identify Me If You Can	1,819	23,236	23,237
7042	Text Normalization Challenge - English Language	1,834	10,886	1,077,678
7115	Cdiscount's Image Classification Challenge	5,859	530,455	1,237,727
7563	Cover Type Prediction of Forests	3,077	6,000	14,000
7634	TensorFlow Speech Recognition Challenge	24,263	3,171	155,365
8307	Data Science Nigeria Telecoms Churn	1,259	180	420
9051	Digit Recongnizer 1438	1,640	2,576	2,575
9491	ML-2018spring-hw5	2,247	100,000	100,000
10851	CSE158 fa18 Category Prediction	3,872	7,000	7,000
11842	AIA image classification by CNN	3,415	1,351	150
12187	CS 4740 Project 4 Revised	1,086	N/A	N/A
12349	2019 1st ML month with KaKR	3,307	209	209
12357	UC Berkeley CS189 HW1 (MNIST)	1,480	3,000	7,000
12358	UC Berkeley CS189 HW1 (SPAM)	1,527	1,757	4,100
12359	UC Berkeley CS189 HW1 (CIFAR-10)	1,336	3,000	7,000
12598	TAU Robot Surface Detection	1,574	852	853
12681	PadhAI: Text - Non Text Classification Level 2	1,185	90	210
13034	UC Berkeley CS189 HW3 (MNIST)	1,193	3,000	7,000
13036	UC Berkeley CS189 HW3 (SPAM)	1,080	1,757	4,100
13078	Homework 2 Part 2 - Classification	2,245	N/A	N/A
13383	ML2019spring-hw2	2,802	8,141	8,141

4.8.1.2 Accuracy: outlier competitions

Figure 4.7 shows the plots corresponding to the three levels of overfitting analysis for the mean accuracy difference outliers discussed in Section 4.4.5.

By reading the Kaggle forums and the description of the dataset construction, we were able to determine a possible cause of overfitting for most of the outlier competitions. In some cases, the competition hosts did not create the public and private test splits in a truly i.i.d. manner, and competitors were able to exploit the difference in distribution. In other cases, the size of the test set was extremely small. The list of outlier competitions and corresponding cause of overfitting follow. We also include links to relevant descriptions of the data or discussions on the Kaggle forums.

- **Competition 12598: TAU Robot Surface Detection** The dataset is a time series that consists of measurements of acceleration, velocity and orientation. In some of the measurement locations, the floor has a slope that makes the orientation channels informative (though the orientation terms are designed to not be informative for the particular task). In order to mitigate this effect, the organizers split the time series data into shorter subsequences and assigned separate subsequences to the public and private split. Thus, the public and private data were not created from a truly i.i.d. split. [3]

- **Competition 12349: 2019 1st ML month with KaKR** The competition uses a small test set consisting of 418 examples (209 in the private test set and 209 in the public test set.)

- **Competition 5903: Prediction Reviews Sentiment Analysis (Light)** The competition uses a small test set consisting of 500 examples (400 in the public test set and 100 in the private test set).

- **Competition 3641: DecMeg2014 - Decoding the Human Brain** The test set contains data generated from 7 subjects and the public and private splits were created in a non-i.i.d. manner using data from individual subjects. `https://www.kaggle.com/c/decoding-the-human-brain/data`

- **Competition 12681: PadhAI: Text - Non Text Classification Level 2** The competition uses a small test set consisting of 300 examples (90 in the public test set and 210 in the private test set).

[3] Personal communication with the competition organizer Heikki Huttunen.

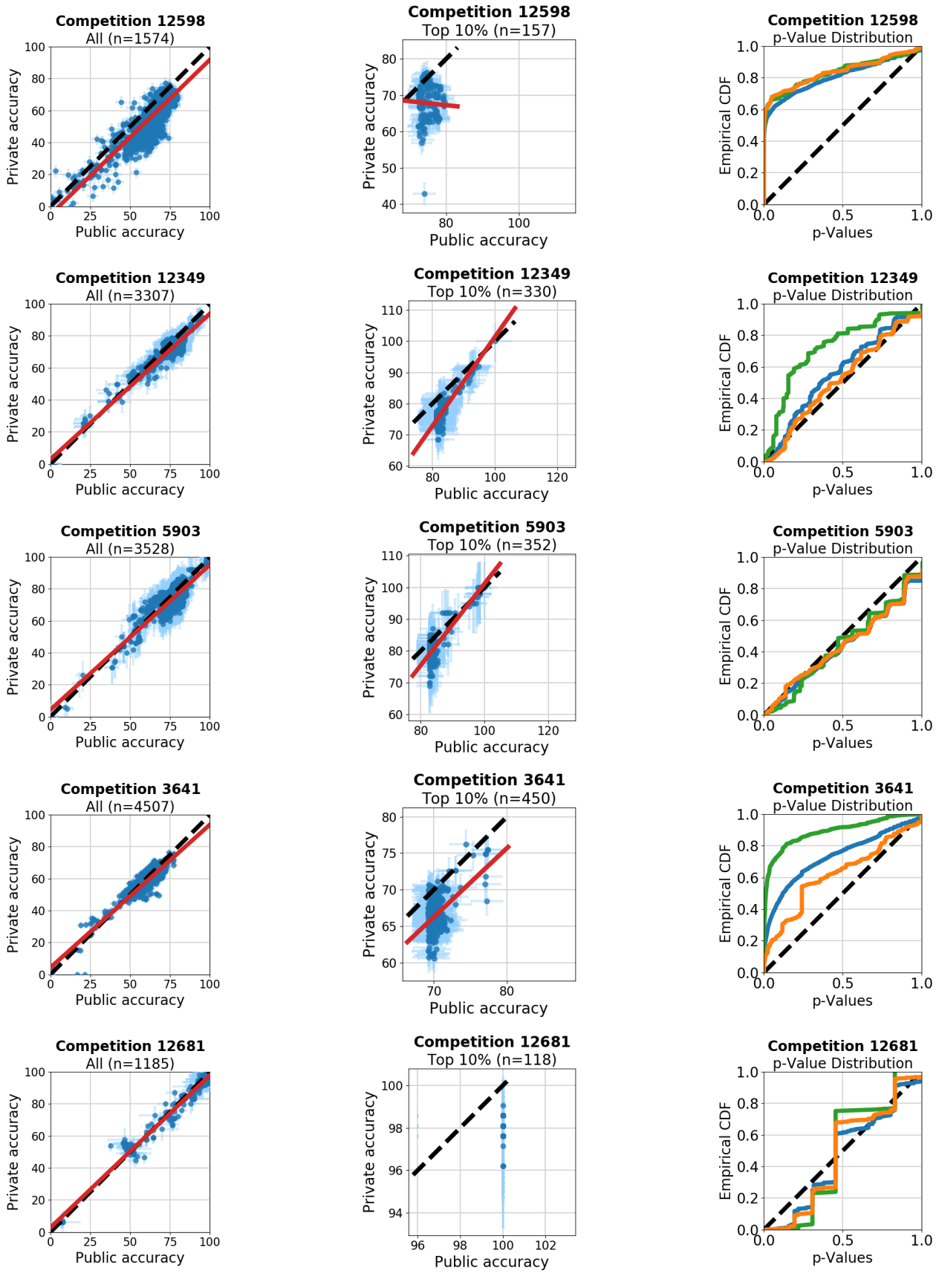


Figure 4.7: Private versus public accuracy for all submissions (left) and the top 10% of submissions

4.8.1.3 Accuracy: all submissions

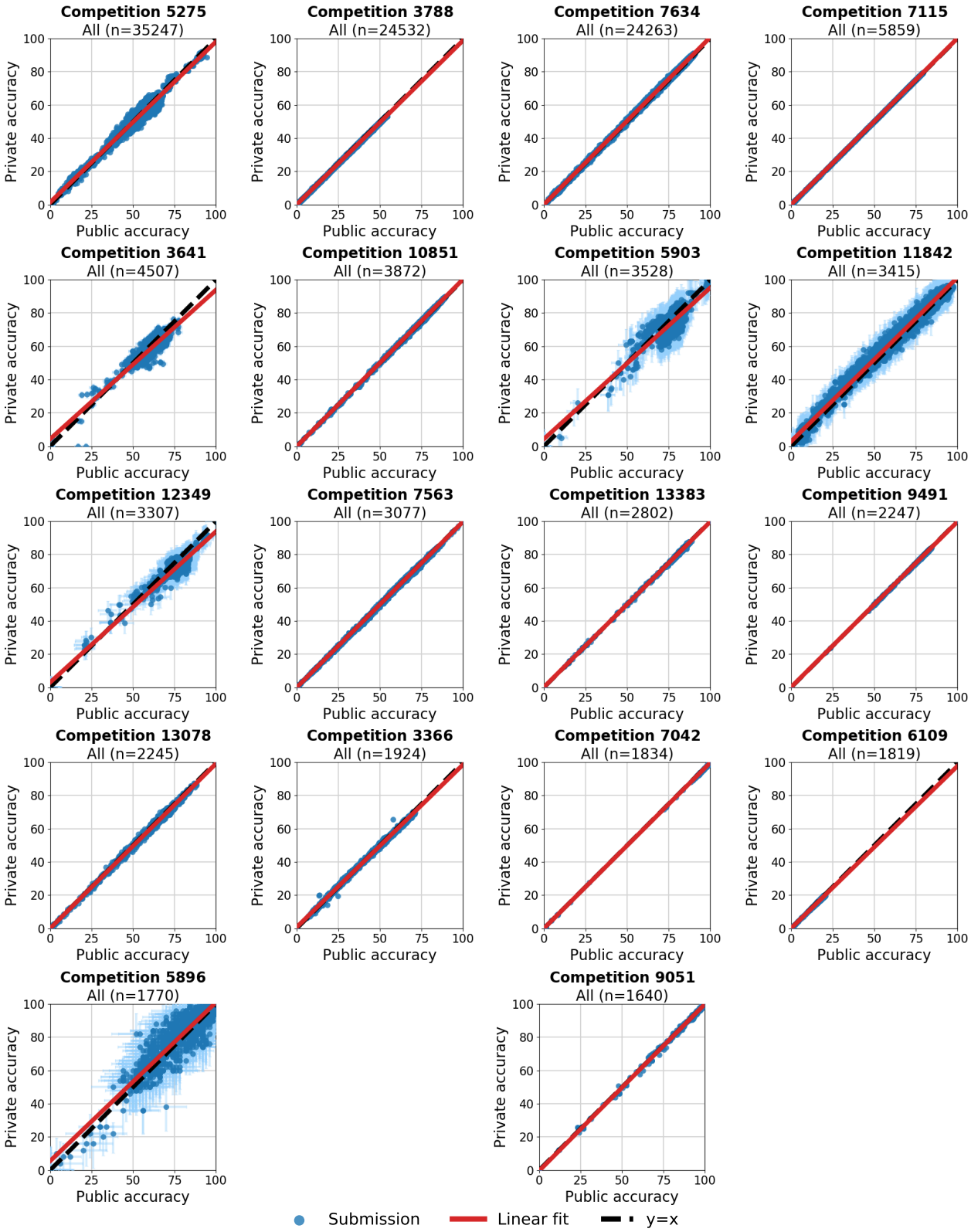


Figure 4.8: Private versus public accuracy for all submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

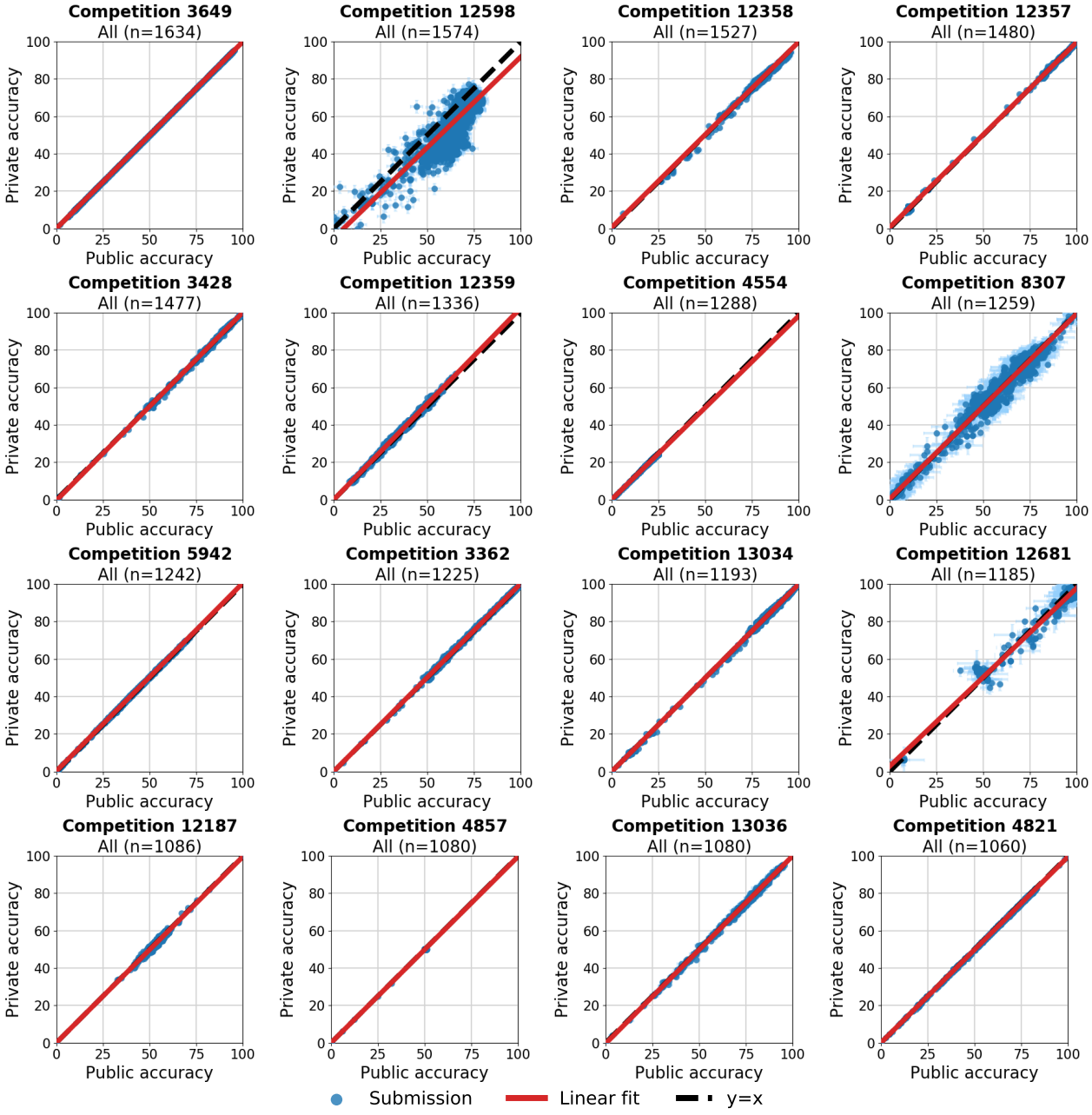


Figure 4.8: Private versus public accuracy for all submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

4.8.1.4 Accuracy: top 10% of submissions

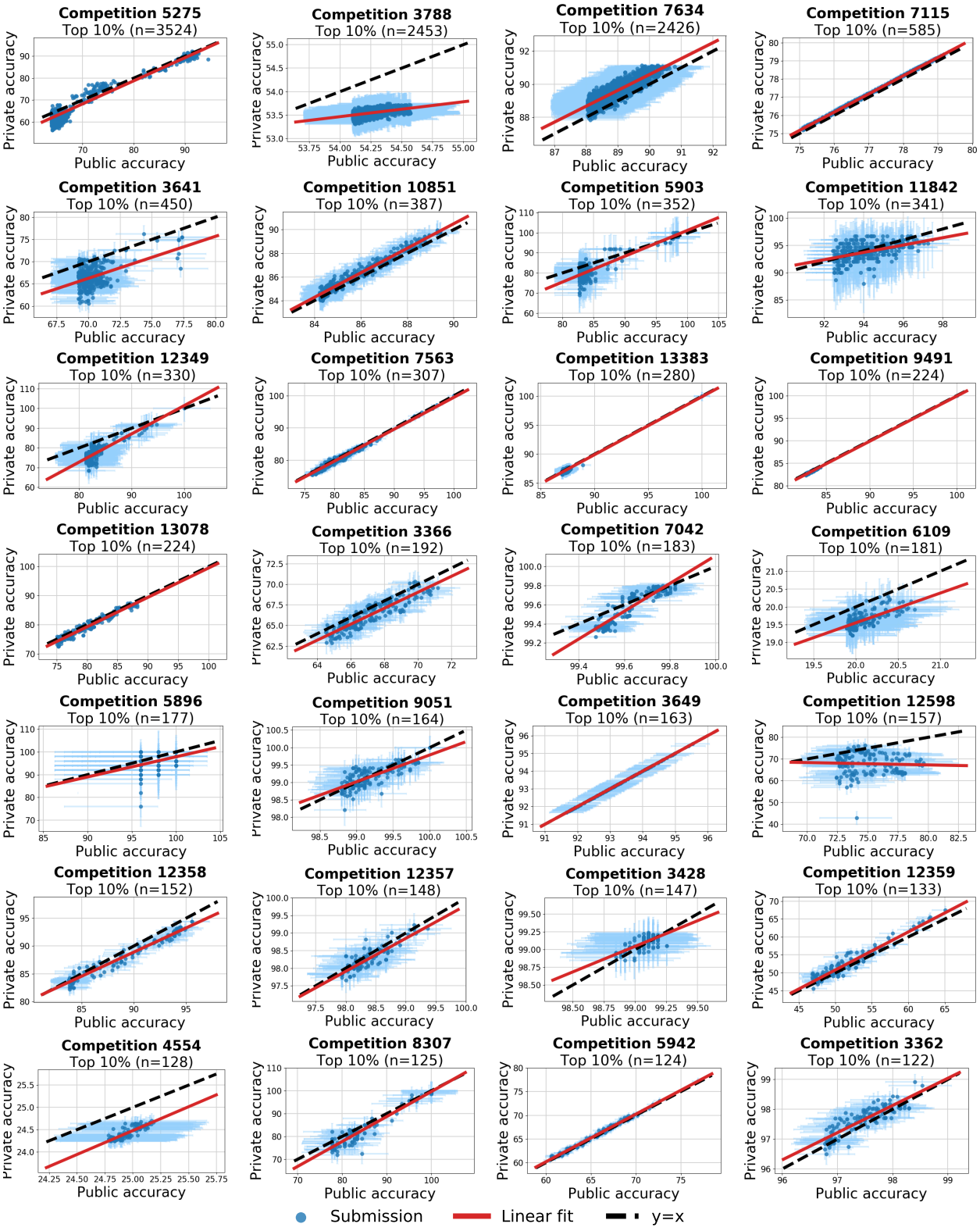


Figure 4.9: Private versus public accuracy for top 10% of submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

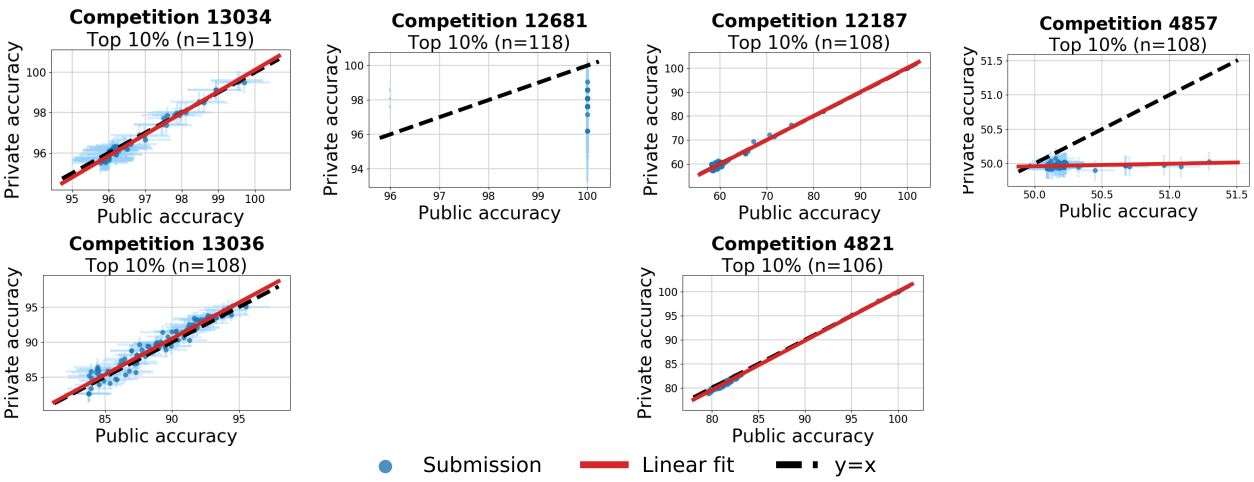


Figure 4.9: Private versus public accuracy for all submissions for the most popular Kaggle accuracy competitions. Each point corresponds to an individual submission (shown with 95% Clopper-Pearson confidence intervals).

4.8.1.5 Accuracy: p-value plots

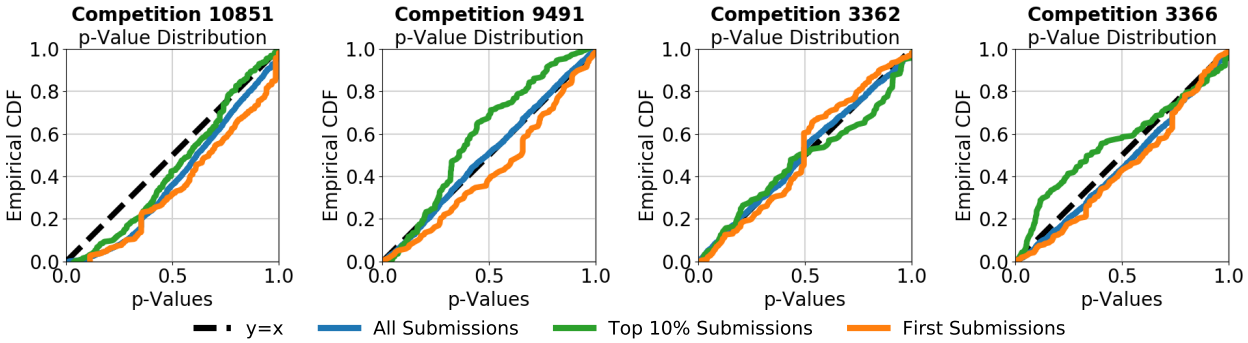


Figure 4.10: Competitions whose empirical CDFs agree with the idealized null model that assumes no overfitting (full details in Section 4.4.3).

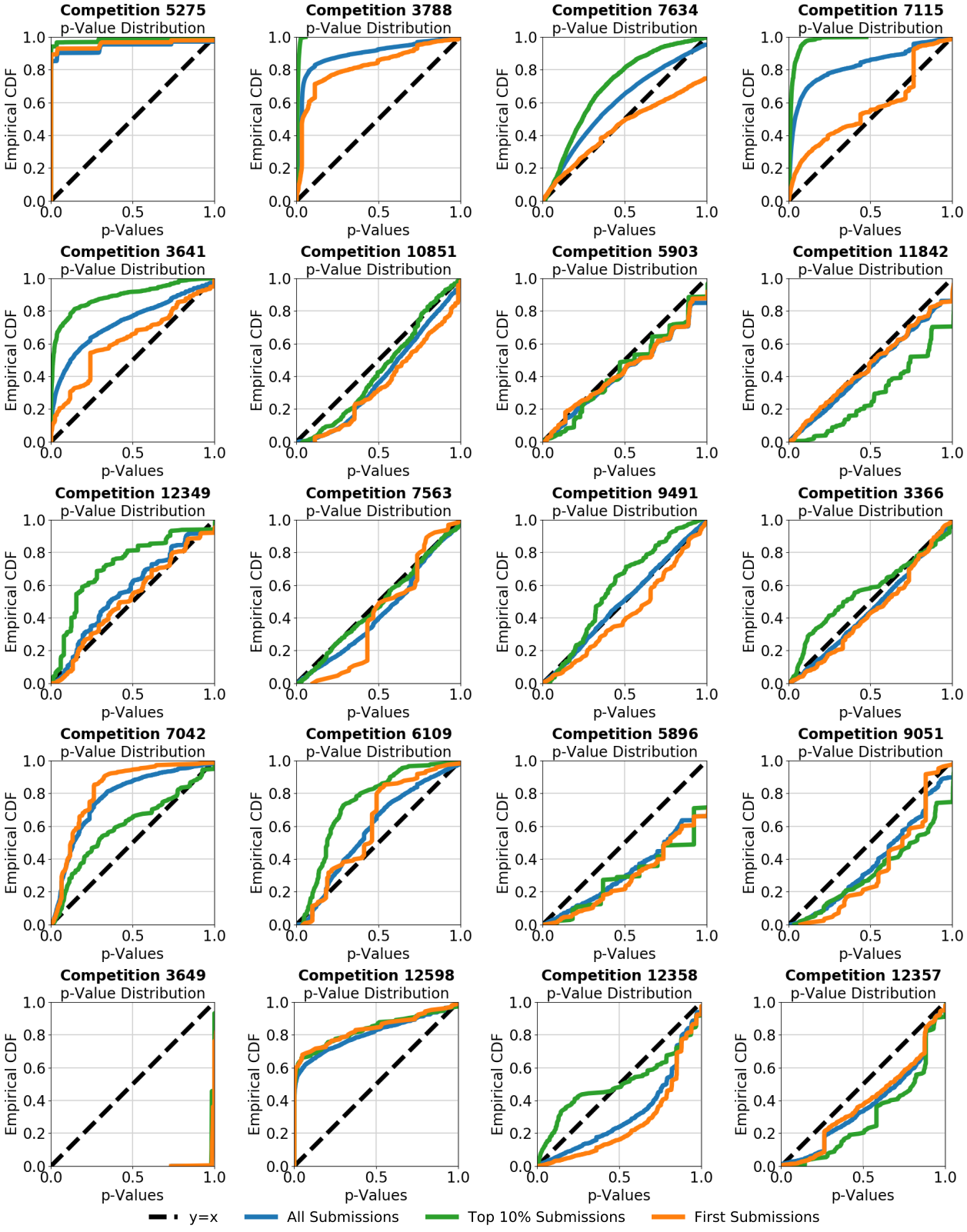


Figure 4.11: Empirical CDFs for an idealized null model that assumes no overfitting for all Kaggle accuracy competitions (full details in Section 4.4.3)

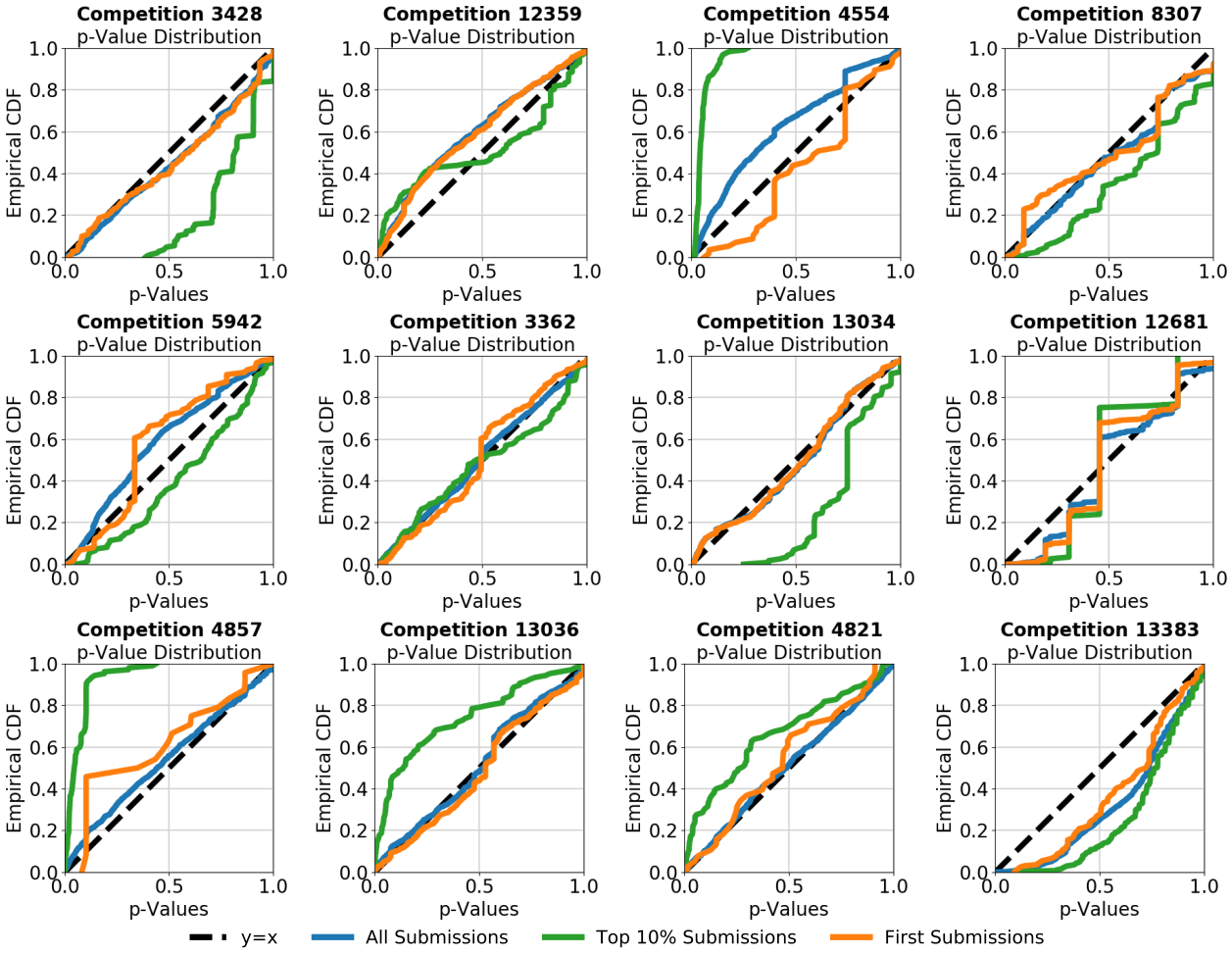


Figure 4.11: Empirical CDFs for an idealized null model that assumes no overfitting for all Kaggle accuracy competitions (full details in Section 4.4.3)

4.8.2 AUC

4.8.2.1 AUC: competition info

Table 4.3: Competitions scored with AUC with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes. A * after the competition name means the name was slightly edited to fit into the table.

AUC				
ID	Name	# Sub.	n_{public}	n_{private}
2439	INFORMS Data Mining Contest 2010	1,483	254	2,285
2445	Predict Grant Applications	2,800	544	1,632
2464	IJCNN Social Network Challenge	1,124	1,792	7,168
2478	Stay Alert! The Ford Challenge	1,402	36,252	84,588
2489	Don't Overfit!	3,775	1,975	17,775
2551	Give Me Some Credit	7,724	30,451	71,052
3338	Amazon.com - Employee Access Challenge	16,896	17,676	41,245
3353	Marinexplore and Cornell Whale Detection*	3,295	16,351	38,152
3469	Influencers in Social Networks	2,109	2,976	2,976
3509	Whale Challenge - Right Whale Redux*	1,005	7,641	17,828
3524	Accelerometer Biometric Competition	7,130	8,102,160	18,905,040
3526	StumbleUpon Evergreen Classification Challenge	7,509	634	2,537
3774	CONNECTOMICS	1,458	75,742	75,743
3897	Acquire Valued Shoppers Challenge	25,205	N/A	N/A
3926	Predicting Excitement at DonorsChoose.org*	12,530	11,193	33,579
3933	MLSP 2014 Schizophrenia Classification Challenge	2,246	0	119,749
3960	American Epilepsy Society Seizure Prediction*	17,782	1,574	2,361
4031	Driver Telematics Analysis	36,072	N/A	N/A
4043	BCI Challenge @ NER 2015	4,348	680	2,721
4294	Facebook Recruiting IV: Human or Robot?	13,559	1,410	3,290
4366	West Nile Virus Prediction	29,963	2,326	113,967
4487	Springleaf Marketing Response	39,439	43,570	101,662
4493	Truly Native?	3,224	26,709	40,064
4657	Homesite Quote Conversion	36,387	52,151	121,685
4986	Santander Customer Satisfaction	93,584	37,909	37,909
5167	PRED 411-2016_04-U2-INSURANCE-A*	1,436	1,070	1,071
5174	Avito Duplicate Ads Detection	8,157	657,602	657,603
5261	Predicting Red Hat Business Value	33,696	149,606	349,081
5390	Melbourne Univ. Seizure Prediction*	10,083	N/A	N/A
6242	Catch Me If You Can: Intruder Detection*	2,479	41,398	41,399
7162	WSDM - KKBox's Music Recommendation*	15,555	1,278,396	1,278,395
8227	2018 Spring CSE6250 HW1	1,170	244,173	244,173

Table 4.3: Competitions scored with AUC with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes. A * after the competition name means the name was slightly edited to fit into the table.

AUC				
ID	Name	# Sub.	n_{public}	n_{private}
8540	AdTracking Fraud Detection*	68,594	3,382,284	15,408,185
9120	Home Credit Default Risk	132,097	9,749	38,995
10683	Microsoft Malware Prediction	43,702	4,947,549	2,905,704
11803	BGU - Machine Learning	2,079	6,849	5,603
11848	Histopathologic Cancer Detection	20,352	28,154	29,304
12512	2019 Spring CSE6250 BDH	1,396	244,173	244,173
12558	WiDS Datathon 2019	3,021	4,312	2,222
12904	Caltech CS 155 2019 Part 1	1,078	1,264,122	1,264,122

4.8.2.2 AUC: outlier competitions

For AUC, we also investigated competitions whose linear fit was a bad approximation to the diagonal $y = x$ line. We first identified these competitions using visual inspection of the plots in Section 4.8.2.3.

By reading the Kaggle forums and the description of the dataset construction, we were able to determine a possible cause of overfitting for most of the outlier competitions. In most cases, the competition hosts did not create the public and private test splits in a truly i.i.d. manner, and competitiors were able to exploit the difference in distribution. In other cases, the size of the test set was extremely small. The list of outlier competitions and corresponding cause of overfitting follow. We also include links to relevant descriptions of the data or discussions on the Kaggle forums.

- **Competition 4043: BCI Challenge @ NER 2015**: The test set contains data generated from 10 subjects and the public and private splits were created in a non-i.i.d. manner using data from individual subjects `https://www.kaggle.com/c/inria-bci-challenge/discussion/12613#latest-65652`.

- **Competition 3926: KDD Cup 2014 - Predicting Excitement at DonorsChoose.org**: Public and private splits are not i.i.d. and likely were created by separating donations projects by the date they were proposed. `https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose/discussion/9772#latest-50827`

- **Competition 3933: MLSP 2014 Schizophrenia Classification Challenge** The test set contains data generated from 58 subjects and the public and private splits were created in a non-i.i.d. manner using data from individual subjects. `https://www.kaggle.com/c/mlsp-2014-mri/discussion/10135#latest-54483`

- **Competition 3774: CONNECTOMICS** Participants knew which test data belonged in the public and private splits because the splits were provided in separate files. `https://www.kaggle.com/c/connectomics/data`

- **Competition 8540: TalkingData AdTracking Fraud Detection Challenge** A second, larger test set that was unintentionally released at the start of the competition and then participants were permitted but not required to use the data. `https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection/discussion/52658#latest-315882`

- **Competition 10683: Microsoft Malware Prediction** The private test data included several severe outliers not present in the public test data, indicating that the public and private split are not i.i.d. `https://www.kaggle.com/c/microsoft-malware-prediction/discussion/84745`

4.8.2.3 AUC: all submissions

Figure 4.12 shows the private versus public AUC scatter plots for each AUC competition with over 1000 submissions in the MetaKaggle dataset.

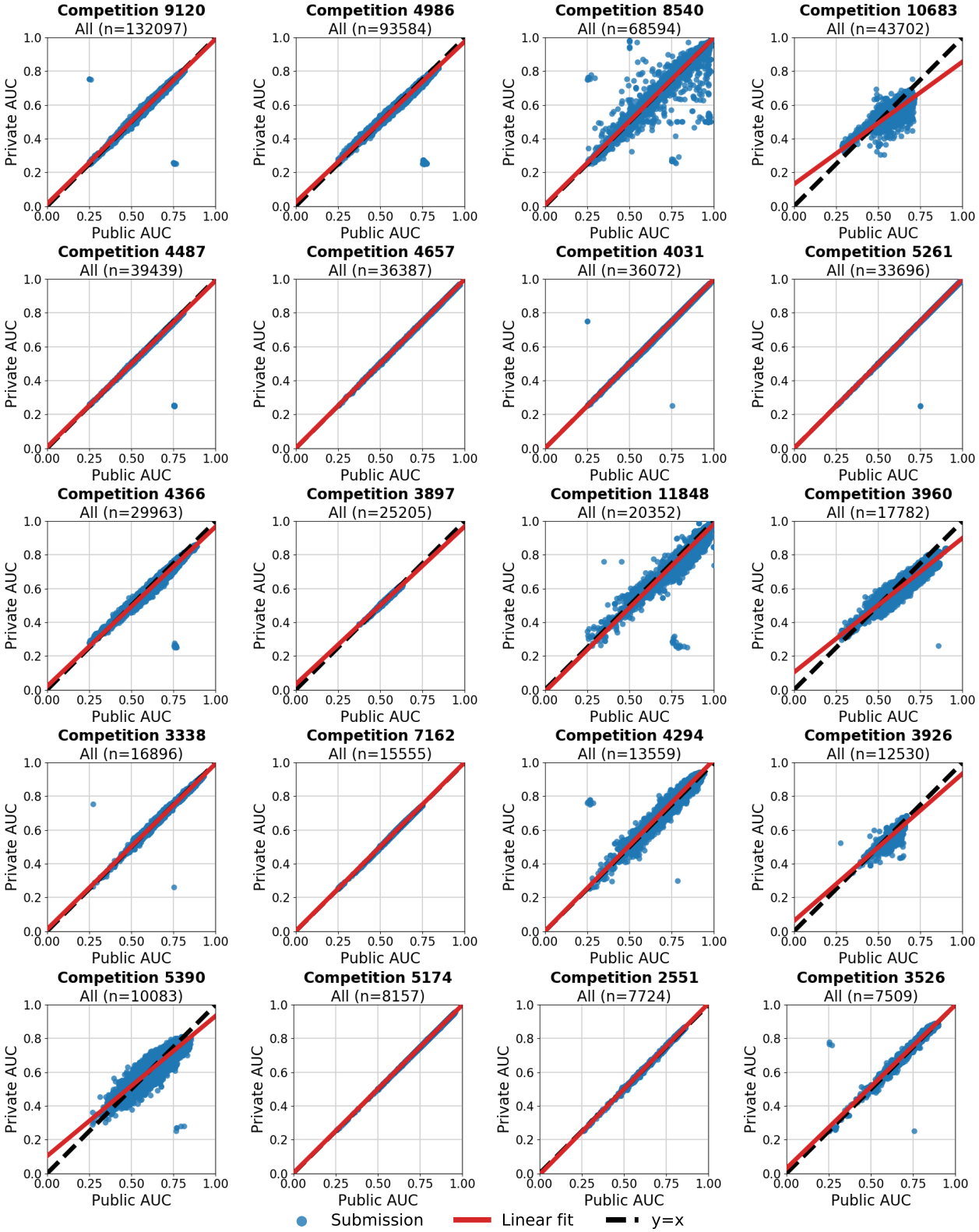


Figure 4.12: Private versus public AUC for all submissions for Kaggle AUC competitions.

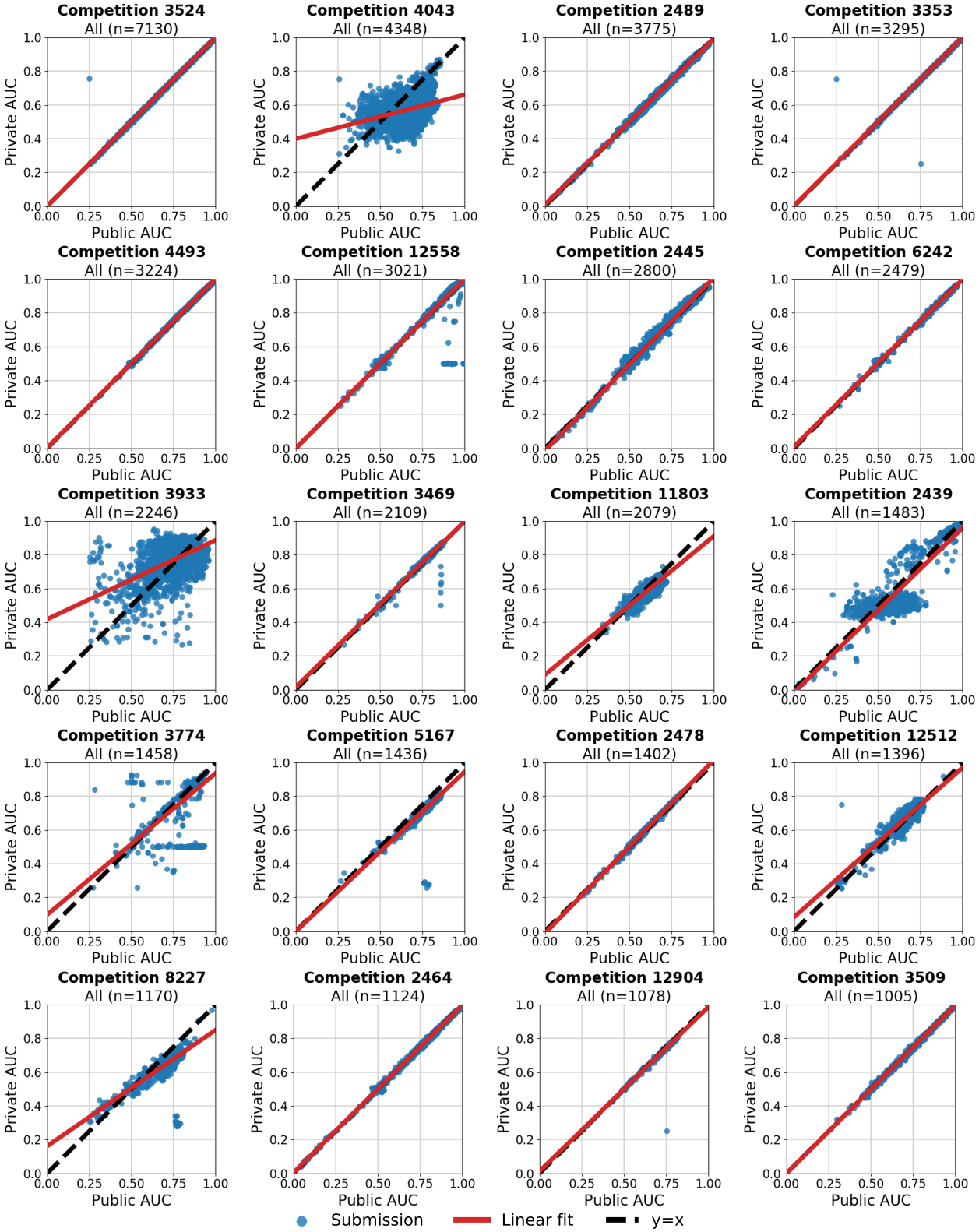


Figure 4.12: Private versus public AUC for all submissions for Kaggle AUC competitions.

4.8.2.4 AUC: top 10% of submissions

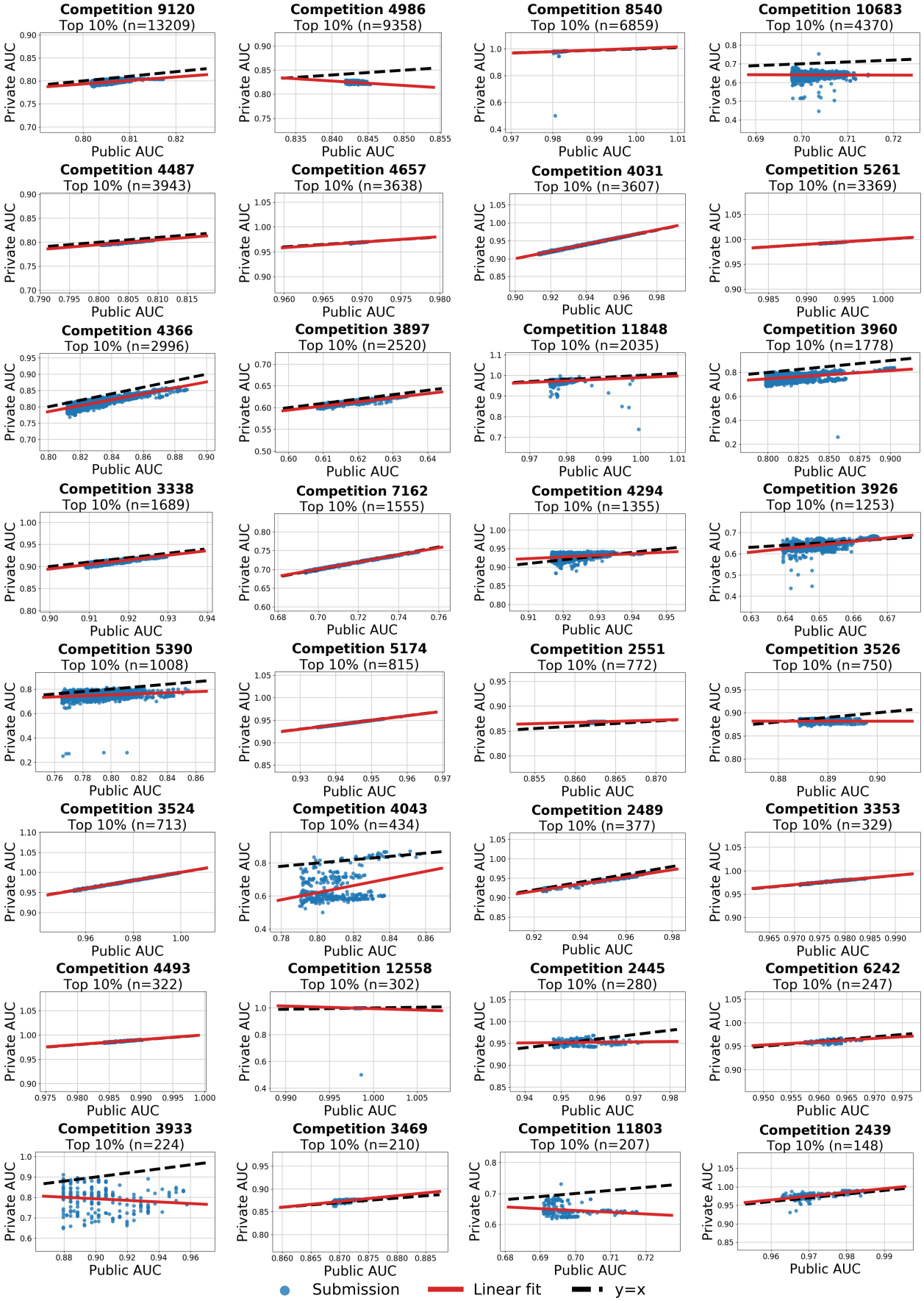


Figure 4.13: Private versus public accuracy for top 10% of submissions for Kaggle AUC

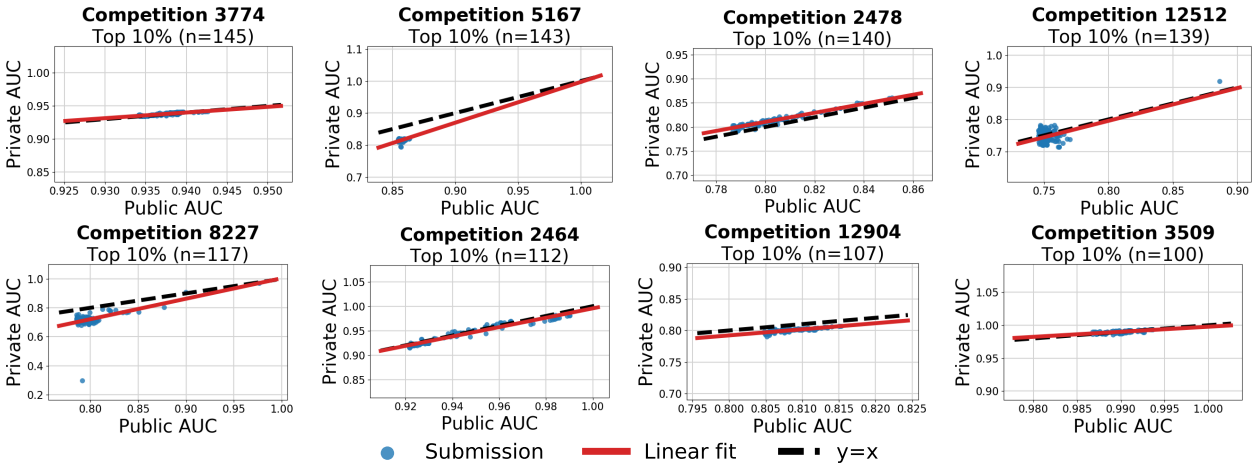


Figure 4.13: Private versus public AUC for top 10% of submissions for Kaggle AUC competitions.

4.8.3 Map@K

4.8.3.1 Map@K: competition info

Table 4.4: Competitions scored with MAP@K with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes.

MAP@K				
ID	Name	# Sub.	n_{public}	n_{private}
4481	Coupon Purchase Prediction	18,487	650	1,515
5056	Expedia Hotel Recommendations	22,713	834,321	1,693,923
5186	Facebook V: Predicting Check Ins	15,127	4,445,370	4,445,369
5497	Outbrain Click Prediction	6,654	9,667,549	22,557,613
5558	Santander Product Recommendation	28,773	278,884	650,731
6818	Humpback Whale Identification	37,529	1,592	6,368
7666	CS5785 Fall 2017 Final Exam	3,898	1,000	1,000
8396	Google Landmark Retrieval Challenge	3,123	0	117,704
8857	Recipient prediction 2018	1,154	1,000	1,000
8900	Freesound General-Purpose Audio Tagging Challenge	5,684	282	9,118
10200	Quick, Draw! Doodle Recognition Challenge	21,407	10,098	102,101
12088	CS5785 Fall 2018 Final	1,877	1,000	1,000

4.8.3.2 Map@K: outlier competitions

We did not observe any MAP@K competitions which exhibit a linear fit that poorly approximates the $y = x$ diagonal line (see Section 4.8.3.3).

4.8.3.3 Map@K: all submissions

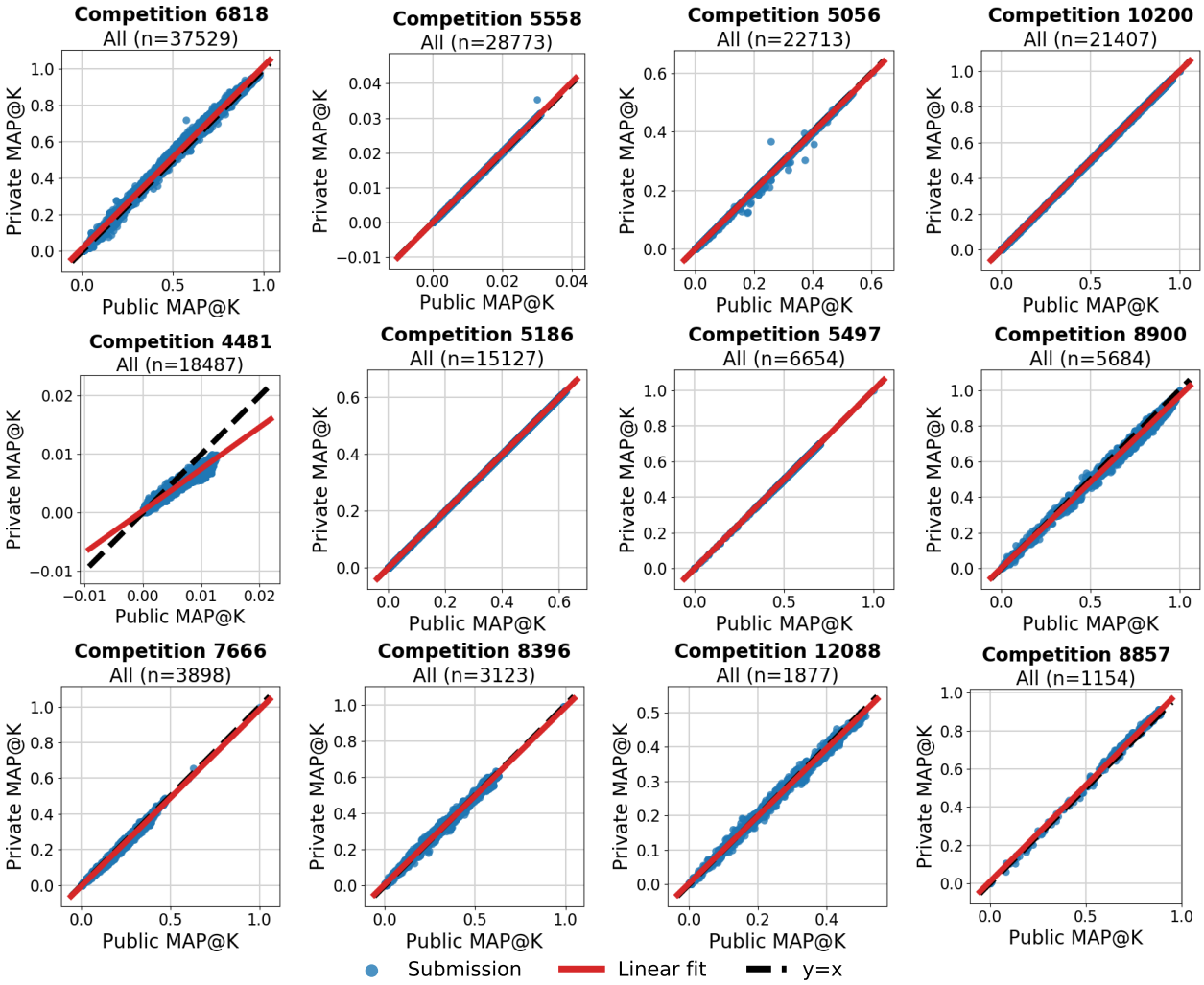


Figure 4.14: Private versus public MAP@K for all submissions for Kaggle MAP@K competitions.

4.8.3.4 Map@K: top 10% of submissions

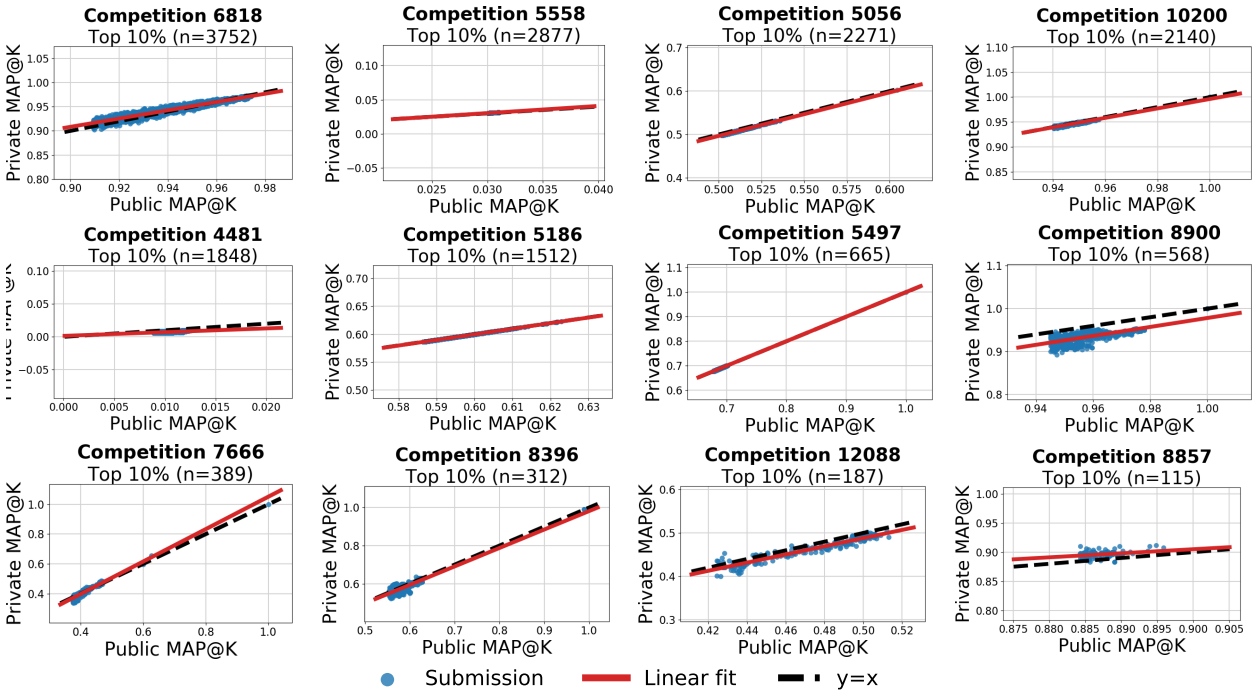


Figure 4.15: Private versus public MAP@K for top 10% of submissions for Kaggle MAP@K competitions.

4.8.4 MulticlassLoss

4.8.4.1 MulticlassLoss: competition info

Table 4.5: Competitions scored with MulticlassLoss with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes.

MulticlassLoss				
ID	Name	# Sub.	n_{public}	n_{private}
4521	Right Whale Recognition	4,789	N/A	N/A
4588	Telstra Network Disruptions	19,603	3,240	7,931
4654	Walmart Recruiting: Trip Type Classification	13,135	18,629	43,467
5048	State Farm Distracted Driver Detection	25,591	N/A	N/A
5340	TalkingData Mobile User Demographics	24,631	33,621	78,450
5568	The Nature Conservancy Fisheries Monitoring	2,100	N/A	N/A
5590	Two Sigma Connect: Rental Listing Inquiries	47,044	N/A	N/A
6243	Intel & MobileODT Cervical Cancer Screening	1,367	N/A	N/A
6841	Personalized Medicine: Redefining Cancer Treatment	3,056	N/A	N/A
7516	Spooky Author Identification	10,640	2,518	5,874

4.8.4.2 MulticlassLoss: outlier competitions

For MulticlassLoss, we investigated competitions whose linear fit was a bad approximation to the diagonal $y = x$ line. We first identified these competitions using visual inspection of the plots in Section 4.8.4.3.

By reading the Kaggle forums and the description of the dataset construction, we were able to determine a possible cause of overfitting for most of the outlier competitions. In most cases, the competition hosts did not create the public and private test splits in a truly i.i.d. manner, and competitiors were able to exploit the difference in distribution. In other cases, the size of the test set was extremely small. The list of outlier competitions and corresponding cause of overfitting follow. We also include links to relevant descriptions of the data or discussions on the Kaggle forums.

- **Competition 6243: Intel & MobileODT Cervical Cancer Screening** The test set was released in two stages. `https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening/data`
- **Competition 6841: Personalized Medicine: Redefining Cancer Treatment** The test set was released in two stages because a team discovered the test data from

Stage 1, including labels, on a public website. https://www.kaggle.com/c/msk-redefining-cancer-treatment/discussion/36383#latest-223033

- **Competition 5568: The Nature Conservancy Fisheries Monitoring** The test set was released in two stages. https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring/data

4.8.4.3 MulticlassLoss: all submissions

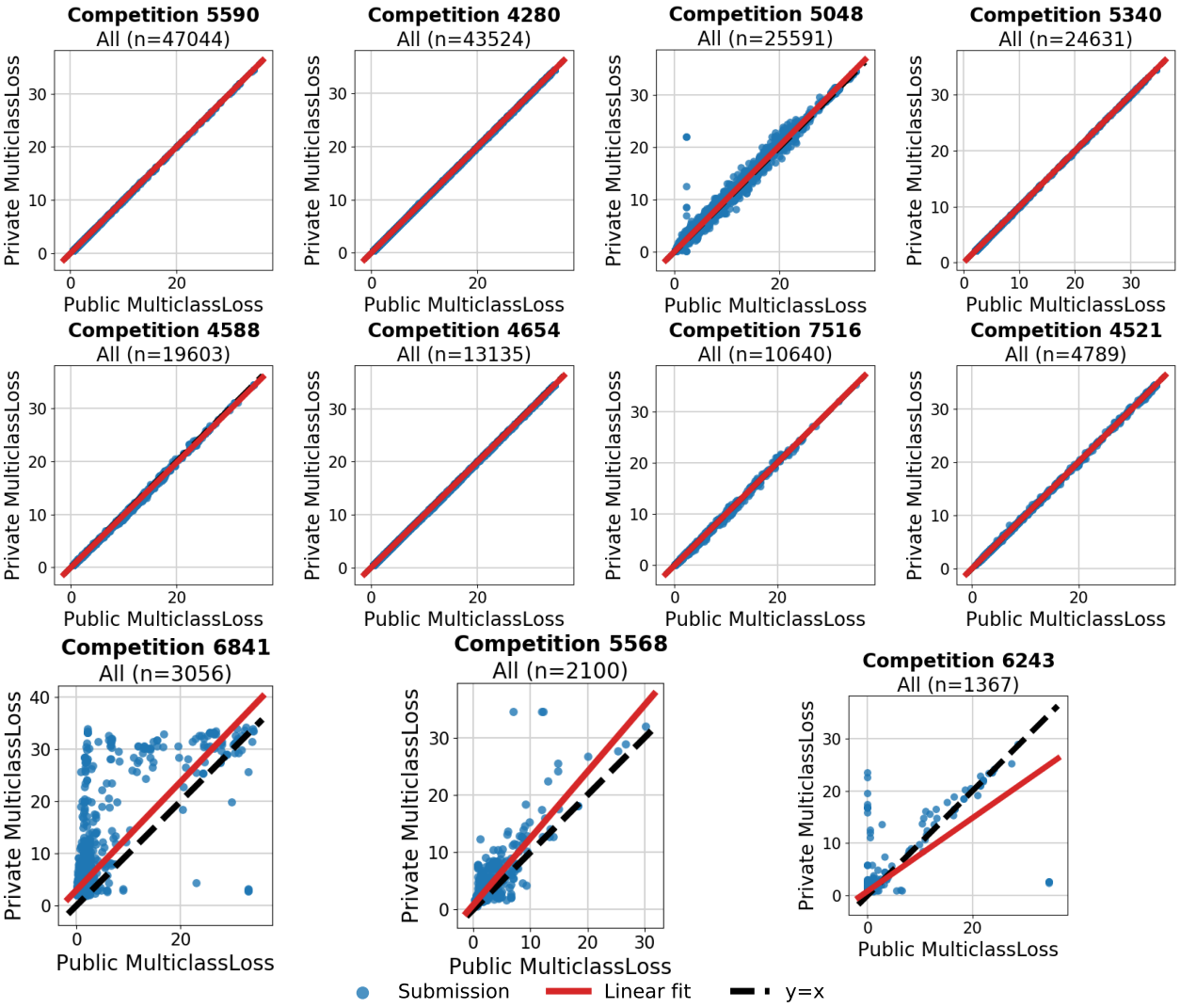


Figure 4.16: Private versus public MulticlassLoss for all submissions for Kaggle MulticlassLoss competitions.

4.8.4.4 MulticlassLoss: top 10% of submissions

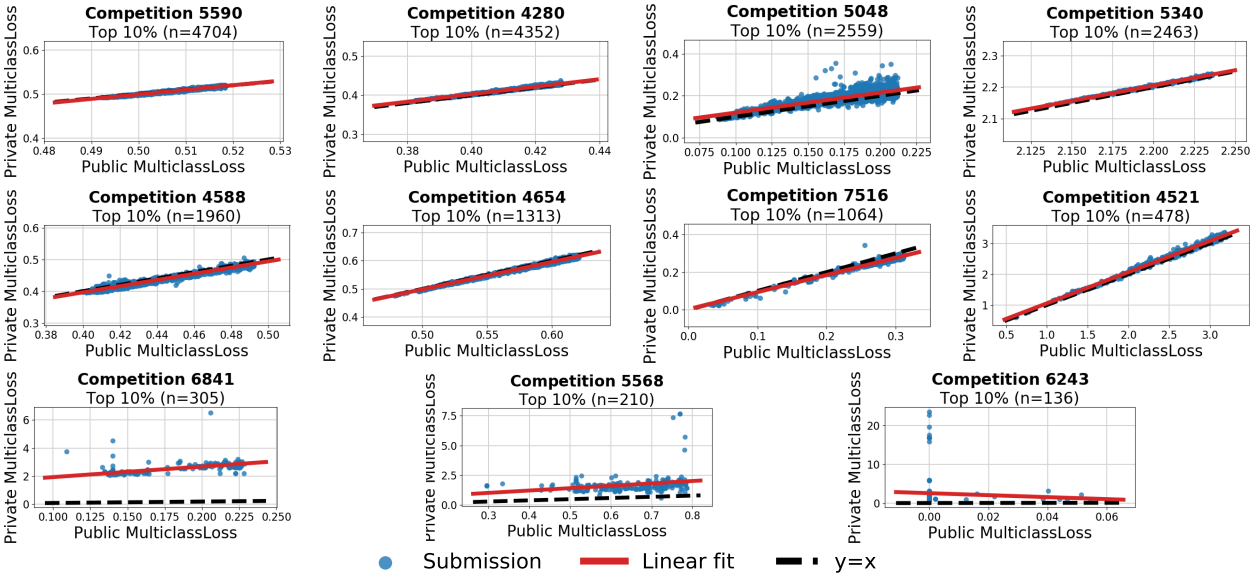


Figure 4.17: Private versus public MulticlassLoss for top 10% of submissions for Kaggle MulticlassLoss competitions.

4.8.5 LogLoss

4.8.5.1 LogLoss: competition info

LogLoss				
ID	Name	# Sub.	n_{public}	n_{private}
2780	Predicting a Biological Response	8,843	625	1,876
2984	Practice Fusion Diabetes Classification	2,200	1,245	3,734
3377	ICDAR2013 - Gender Prediction from Handwriting	1,884	286	486
3934	Display Advertising Challenge	8,640	1,208,427	4,833,708
3984	Tradeshift Text Classification	5,659	163,525	381,557
4120	Click-Through Rate Prediction	31,019	915,493	3,661,972
4438	Avito Context Ad Clicks	5,951	2,344,909	5,471,453
4852	BNP Paribas Cardif Claims Management	54,519	37,750	76,643
4862	March Machine Learning Mania 2016	1,054	N/A	N/A
6004	Data Science Bowl 2017	1,694	N/A	N/A
6277	Quora Question Pairs	53,927	140,748	2,205,048
7163	WSDM - KKBox's Churn Prediction Challenge	6,256	N/A	N/A
7380	Statoil/C-CORE Iceberg Classifier Challenge	42,936	N/A	N/A
7823	Attrition de clientes	1,911	N/A	N/A
8310	Google Cloud & NCAA® ML Competition 2018-Men's	1,660	N/A	N/A

Table 4.6: Competitions scored with LogLoss with greater than 1000 submissions. n_{public} is the size of the public test set and n_{private} is the size of the private test set. N/A means that we could not access the competition data to compute the dataset sizes.

4.8.5.2 LogLoss: outlier competitions

For LogLoss, we observed one competition (Competition 6004: Data Science Bowl 2017) whose linear fit was a bad approximation the diagonal $y = x$ line. The competition shows approximately 13 outliers when looking at the scatter plot for the top 10% of submissions (see Figure 4.19).

The competition is unique in two aspects: first, the competition used two stages with separate test sets released at different times and second, participants were allowed to use external data. Participants were eventually given all the labels to the Stage 1 test data, but the Stage 2 test data was different in distribution from the Stage 1 test data (see `https://www.kaggle.com/c/data-science-bowl-2017/discussion/31383#latest-174302`). It is unclear whether the final private test data was generated from an i.i.d. split with either the Stage 1 test data or the Stage 2 data. Moreover, training the model on external data violates our assumption that the models were only trained and evaluated on i.i.d. data.

4.8.5.3 LogLoss: all submissions

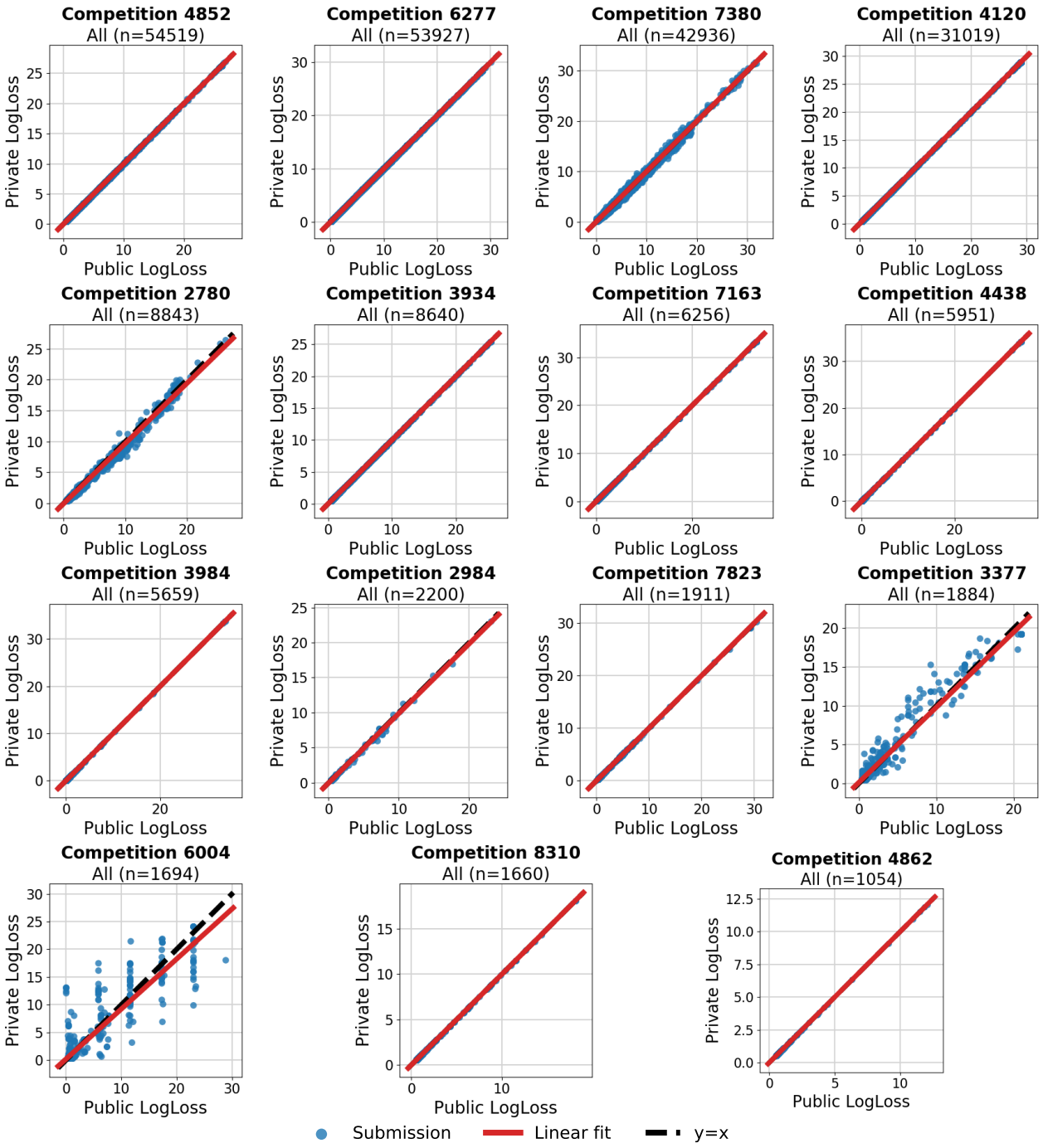


Figure 4.18: Private versus public LogLoss for all submissions for Kaggle LogLoss competitions.

4.8.5.4 LogLoss: top 10% of submissions

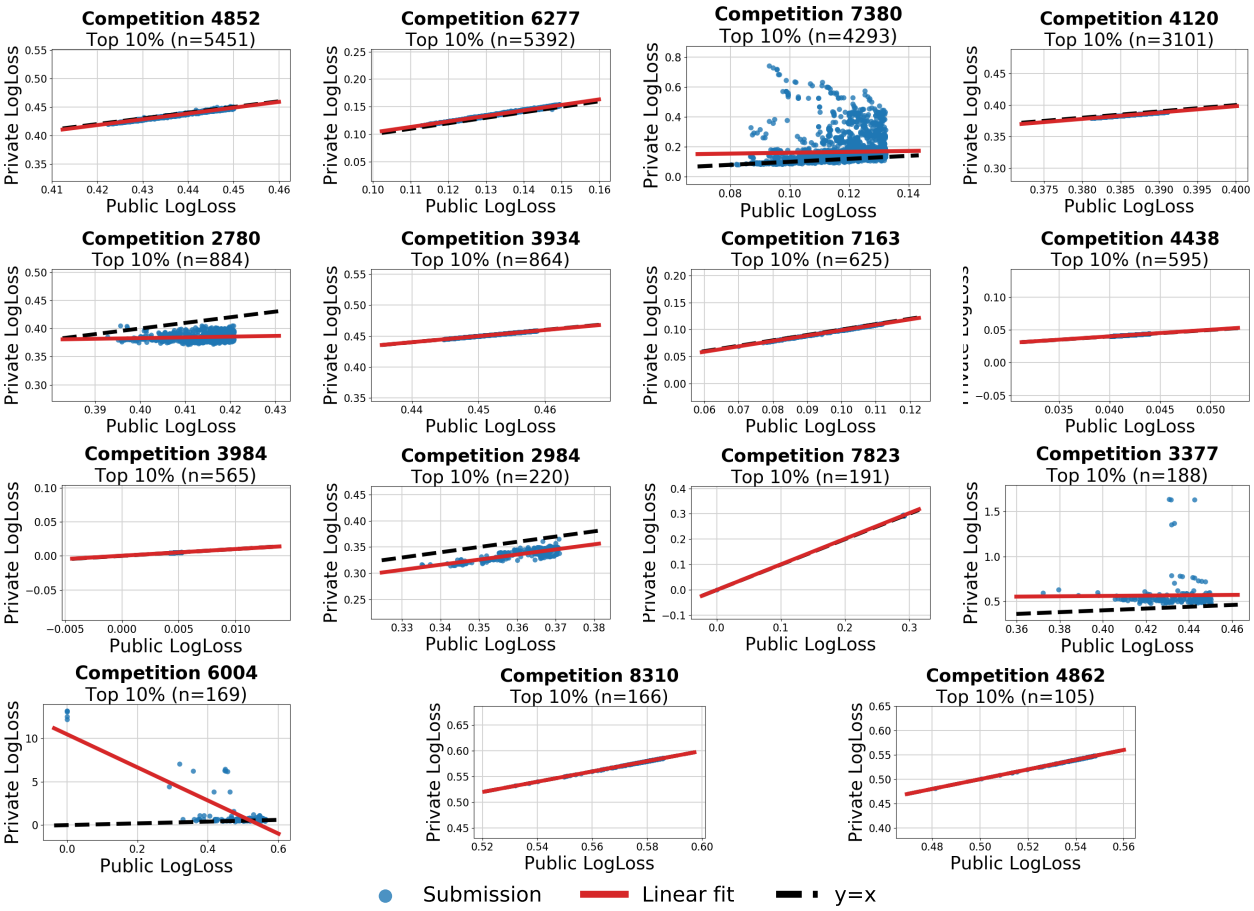


Figure 4.19: Private versus public LogLoss for top 10% of submissions for Kaggle LogLoss competitions.

4.8.6 Mean score differences over time

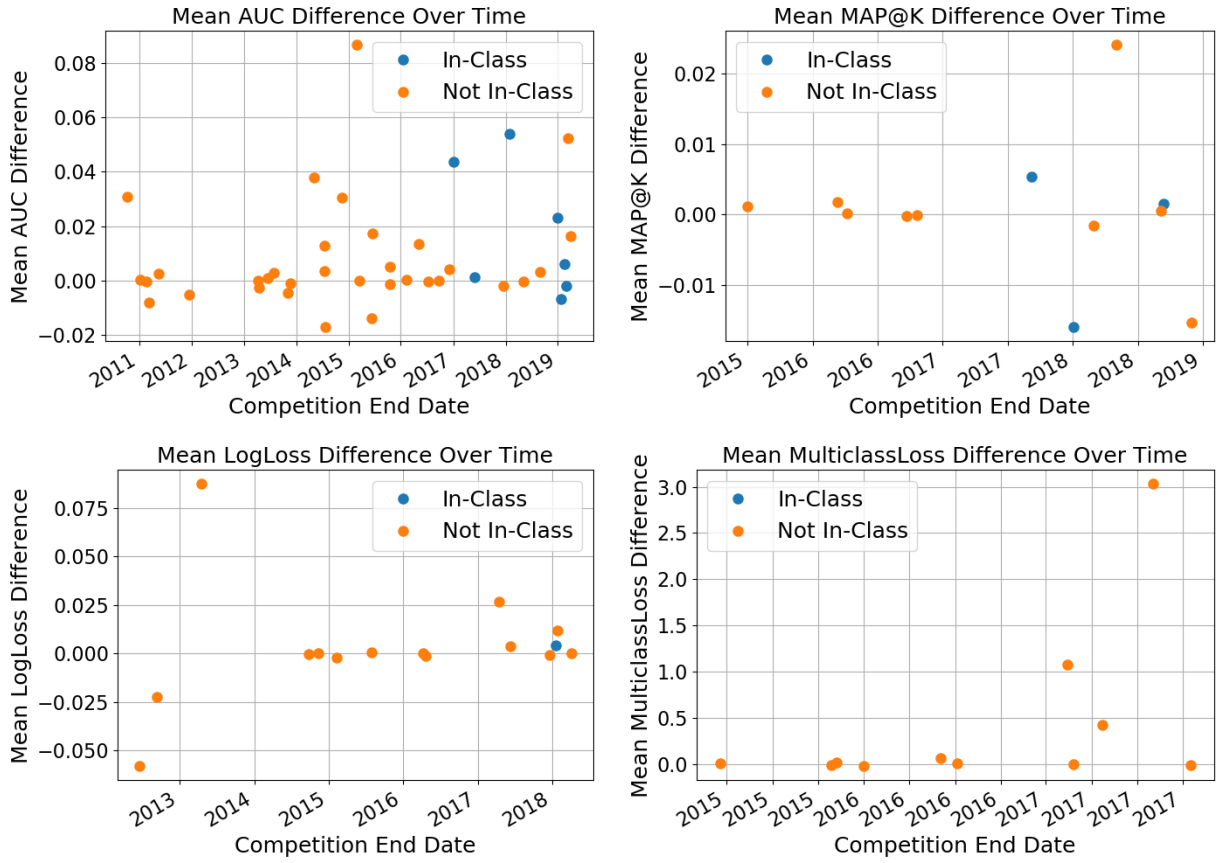


Figure 4.20: Mean score differences (across all pre-deadline submissions) versus competition end date for all classification evaluation metrics with at least 10 competitions with at least 1000 submissions each. Score difference is public minus private for metrics where a higher score is better (AUC, CategorizationAccuracy (included in the main text), and MAP@K) and private minus public for metrics where a lower score is better (LogLoss and MulticlassLoss). Blue dots show InClass competitions and orange dots show all other competition types (Featured, Research, Getting Started, and Playground).

Chapter 5

Conclusion

In this thesis, we empirically evaluated key theoretical pillars of machine learning surrounding generalization and overfitting in order to better understand machine learning robustness. We showed that the choice of optimization algorithm can influence the generalization error of a trained model and demonstrated that adaptive gradient methods frequently find solutions with worse generalization error than stochastic gradient descent. We then turned to the meta question of whether the test error itself is a realiable measurement of generalization error. While the prevailing concern is that test set reuse leads to overfitting and overly optimistic measurements of generalization error, we saw little evidence of overfitting in both machine learning benchmarks and competitions. Instead, we demonstrated that machine learning classifiers are extremely brittle to small, benign distribution shifts. Overall, designing models that still work in dynamic environments remains a key challenge for robust machine learning.

5.1 Future Work

This thesis highlights many opportunities for future work, which we outline below.

Empirically understanding overfitting There are multiple directions for empirically understanding overfitting in more detail. Our analysis of the Meta-Kaggle dataset in Chapter 4 focused on classification competitions, but Kaggle also hosts many regression competitions. Is there more adaptive overfitting in regression? Answering this question will likely require access to the individual predictions of the Kaggle submissions to appropriately handle outlier submissions. In addition, there are still questions among the classification competitions. For instance, one refinement of our analysis here is to obtain statistical measures of uncertainty for competitions evaluated with metrics such as AUC (which will also require a more fine-grained version of the Kaggle data). Finally, another important question is whether other competition platforms such as CodaLab [11] or EvalAI [97] also show few signs of adaptive overfitting.

Measuring Human Accuracy. One interesting question is whether our new test sets are also harder for humans. As a first step in this direction, our human accuracy experiment on CIFAR-10 (see Section 3.5.2.5) shows that average human performance is not affected significantly by the distribution shift between the original and new images that are most difficult for the models. This suggests that the images are only harder for the trained models and not for humans. But a more comprehensive understanding of the human baseline will require additional human accuracy experiments on both CIFAR-10 and ImageNet.

Characterizing the Distribution Gap. Why do the classification models in our ImageNet and CIFAR-10 testbed perform worse on the new test sets? The selection frequency experiments in Section 3.4 suggest that images selected less frequently by the MTurk workers are harder for the models. However, the selection frequency analysis does not explain what aspects of the images make them harder. Candidate hypotheses are object size, special filters applied to the images (black & white, sepia, etc.), or unusual object poses. Exploring whether there is a succinct description of the difference between the original and new test distributions is an interesting direction for future work.

Learning More Robust Models. An overarching goal is to make classification models more robust to small variations in the data. If the change from the original to our new test sets can be characterized accurately, techniques such as data augmentation or robust optimization may be able to close some of the accuracy gap. Otherwise, one possible approach could be to gather particularly hard examples from Flickr or other data sources and expand the training set this way. However, it may also be necessary to develop entirely novel approaches to image classification.

Building Further Test Sets. The dominant paradigm in machine learning is to evaluate the performance of a classification model on a single test set per benchmark. Our results suggest that this is not comprehensive enough to characterize the reliability of current models. To understand their generalization abilities more accurately, new test data from various sources may be needed. One intriguing question here is whether accuracy on other test sets will also follow a linear function of the original test accuracy.

Suggestions For Future Datasets. We found that it is surprisingly difficult to create a new test set that matches the distribution of an existing dataset. Based on our experience with this process, we provide some suggestions for improving machine learning datasets in the future:

- **Code release.** It is hard to fully document the dataset creation process in a paper because it involves a long list of design choices. Hence it would be beneficial for reproducibility efforts if future dataset papers released not only the data but also all code used to create the datasets.

- **Annotator quality.** Our results show that changes in the human annotation process can have significant impact on the difficulty of the resulting datasets. To better understand the quality of human annotations, it would be valuable if authors conducted a standardized test with their annotators (e.g., classifying a common set of images) and included the results in the description of the dataset. Moreover, building variants of the test set with different annotation processes could also shed light on the variability arising from this data cleaning step.

- **"Super hold-out".** Having access to data from the original CIFAR-10 and ImageNet data collection could have clarified the cause of the accuracy drops in our experiments. By keeping an additional test set hidden for multiple years, future benchmarks could explicitly test for adaptive overfitting after a certain time period.

- **Simpler tasks for humans.** The large number of classes and fine distinctions between them make ImageNet a particularly hard problem for humans without special training. While classifying a large variety of objects with fine-grained distinctions is an important research goal, there are also trade-offs. Often it becomes necessary to rely on images with high annotator agreement to ensure correct labels, which in turn leads to bias by excluding harder images. Moreover, the large number of classes causes difficulties when characterizing human performance. So an alternative approach for a dataset could be to choose a task that is simpler for humans in terms of class structure (fewer classes, clear class boundaries), but contains a larger variety of object poses, lighting conditions, occlusions, image corruptions, etc.

- **Test sets with expert annotations.** Compared to building a full training set, a test set requires a smaller number of human annotations. This makes it possible to employ a separate labeling process for the test set that relies on more costly expert annotations. While this violates the assumption that train and test splits are i.i.d. from the same distribution, the expert labels can also increase quality both in terms of correctness and example diversity.

Finally, we emphasize that our recommendations here should *not* be seen as flaws in CIFAR-10 or ImageNet. Both datasets were assembled in the late 2000s for an accuracy regime that is very different from the state-of-the-art now. Over the past decade, especially ImageNet has successfully guided the field to increasingly better models, thereby clearly demonstrating the immense value of this dataset. But as models have increased in accuracy and our reliability expectations have grown accordingly, it is now time to revisit how we create and utilize datasets in machine learning.

Bibliography

[1] Alex Berg. Personal communication. 2018.

[2] Battista Biggio and Fabio Roli. "Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning". In: *Pattern Recognition* (2018). https://arxiv.org/abs/1712.03141.

[3] Avrim Blum and Moritz Hardt. "The Ladder: A Reliable Leaderboard for Machine Learning Competitions". In: *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1502.04585. 2015.

[4] Avrim Blum and Moritz Hardt. "The Ladder: A Reliable Leaderboard for Machine Learning Competitions". In: *International Conference on Machine Learning (ICML)*. 2015.

[5] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. 2016.

[6] Yunpeng Chen et al. "Dual path networks". In: *Neural Information Processing Systems (NIPS)*. https://arxiv.org/abs/1707.01629. 2017.

[7] Do Kook Choe and Eugene Charniak. "Parsing as Language Modeling". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. Ed. by Jian Su, Xavier Carreras, and Kevin Duh. The Association for Computational Linguistics, 2016, pp. 2331–2336. URL: http://aclweb.org/anthology/D/D16/D16-1257.pdf.

[8] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1610.02357. 2017.

[9] Stephane Clinchant et al. "XRCE's participation to ImagEval". http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.6670&rep=rep1&type=pdf. 2007.

[10] Adam Coates, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. http://proceedings.mlr.press/v15/coates11a.html. 2011.

[11] *CodaLab*. https://competitions.codalab.org/competitions/.

[12] James Cross and Liang Huang. "Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas*. Ed. by Jian Su, Xavier Carreras, and Kevin Duh. The Association for Computational Linguistics, 2016, pp. 1–11. URL: http://aclweb.org/anthology/D/D16/D16-1001.pdf.

[13] Ekin D. Cubuk et al. "AutoAugment: Learning Augmentation Policies from Data". https://arxiv.org/abs/1805.09501. 2018.

[14] Jia Deng. "Large Scale Visual Recognition". ftp://ftp.cs.princeton.edu/techreports/2012/923.pdf. PhD thesis. Princeton University, 2012.

[15] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. http://www.image-net.org/papers/imagenet_cvpr09.pdf. 2009.

[16] Terrance DeVries and Graham W Taylor. "Improved regularization of convolutional neural networks with Cutout". https://arxiv.org/abs/1708.04552. 2017.

[17] Simon S Du et al. "Gradient descent finds global minima of deep neural networks". In: *International Conference on Machine Learning (ICML)*. 2019.

[18] Simon S Du et al. "Gradient descent provably optimizes over-parameterized neural networks". In: (2019).

[19] John C. Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159. URL: http://dl.acm.org/citation.cfm?id=2021068.

[20] Cynthia Dwork et al. "Preserving statistical validity in adaptive data analysis". In: *ACM symposium on Theory of computing (STOC)*. 2015.

[21] Logan Engstrom et al. "A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations". http://arxiv.org/abs/1712.02779. 2017.

[22] Logan Engstrom et al. "Exploring the Landscape of Spatial Robustness". In: *International Conference on Machine Learning (ICML)*. 2019.

[23] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* (2010). http://dx.doi.org/10.1007/s11263-009-0275-4.

[24] Alhussein Fawzi and Pascal Frossard. "Manitest: Are classifiers really invariant?" In: *British Machine Vision Conference (BMVC)*. https://arxiv.org/abs/1507.06535. 2015.

[25] Li Fei-Fei, Rob Fergus, and Pietro Perona. "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *Computer Vision and Image Understanding* (2007). http://dx.doi.org/10.1016/j.cviu.2005.09.012.

[26] Vitaly Feldman, Roy Frostig, and Moritz Hardt. "The advantages of multiple classes for reducing overfitting from test set reuse". In: *International Conference on Machine Learning (ICML)* (2019).

[27] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics*. 1980.

[28] Xavier Gastaldi. "Shake-shake regularization". https://arxiv.org/abs/1705.07485. 2017.

[29] Robert Geirhos et al. "Generalisation in humans and deep neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. http://papers.nips.cc/paper/7982-generalisation-in-humans-and-deep-neural-networks.pdf. 2018.

[30] Ben Hamner. "Popular Datasets Over Time". Accessed: 2019-01-22. 2017.

[31] Dongyoon Han, Jiwhan Kim, and Junmo Kim. "Deep pyramidal residual networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1610.02915. 2017.

[32] Kaiming He et al. "Deep residual learning for image recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1512.03385. 2016.

[33] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1502.01852. 2015.

[34] Kaiming He et al. "Identity mappings in deep residual networks". In: *European Conference on Computer Vision (ECCV)*. https://arxiv.org/abs/1603.05027. 2016.

[35] Dan Hendrycks and Thomas Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1807.01697. 2019.

[36] Sepp Hochreiter and Jürgen Schmidhuber. "Flat minima". In: *Neural Computation* 9.1 (1997), pp. 1–42.

[37] Hossein Hosseini and Radha Poovendran. "Semantic Adversarial Examples". In: *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. https://arxiv.org/abs/1804.00499. 2018.

[38] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". https://arxiv.org/abs/1704.04861. 2017.

[39] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1709.01507. 2018.

[40] Gao Huang et al. "Densely connected convolutional networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1608.06993. 2017.

[41] Forrest N. Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". https://arxiv.org/abs/1602.07360. 2016.

[42] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1502.03167. 2015.

[43] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". arXiv:1611.07004. 2016.

[44] *Kaggle*. https://www.kaggle.com/.

[45] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Geometric Robustness of Deep Networks: Analysis and Improvement". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1711.09115. 2018.

[46] Andrej Karparthy. *A Peek at Trends in Machine Learning*. https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106. Accessed: 2017-05-17.

[47] Andrej Karpathy. "Lessons learned from manually classifying CIFAR-10". http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/. 2011.

[48] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. "Generalization in deep learning". https://arxiv.org/abs/1710.05468. 2017.

[49] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *The International Conference on Learning Representations (ICLR)*. 2017.

[50] D.P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *The International Conference on Learning Representations (ICLR)* (2015).

[51] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. "Do Better ImageNet Models Transfer Better?" http://arxiv.org/abs/1805.08974. 2018.

[52] Ivan Krasin et al. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." https://storage.googleapis.com/openimages/web/index.html. 2017.

[53] Alex Krizhevsky. "Learning multiple layers of features from tiny images". https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. 2009.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks. 2012.

[55] Fei-Fei Li and Jia Deng. "ImageNet: Where have we been? Where are we going?" http://image-net.org/challenges/talks_2017/imagenet_ilsvrc2017_v1.0.pdf. 2017.

[56] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *International Conference on Learning Representations (ICLR)*. 2016.

[57] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)*. https://arxiv.org/abs/1405.0312. 2014.

[58] Chenxi Liu et al. "Progressive neural architecture search". In: *European Conference on Computer Vision (ECCV)*. https://arxiv.org/abs/1712.00559. 2018.

[59] Shuying Liu and Weihong Deng. "Very deep convolutional neural network based image classification using small training sample size". In: *Asian Conference on Pattern Recognition (ACPR)*. https://ieeexplore.ieee.org/document/7486599/. 2015.

[60] Siyuan Ma and Mikhail Belkin. "Diving into the shallows: a computational perspective on large-scale shallow learning". arXiv:1703.10622. 2017.

[61] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1706.06083. 2018.

[62] Jitendra Malik. "Technical Perspective: What Led Computer Vision to Deep Learning?" In: *Communications of the ACM* (2017). http://doi.acm.org/10.1145/3065384.

[63] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. "Building a Large Annotated Corpus of English: The Penn Treebank". In: *COMPUTATIONAL LINGUISTICS* 19.2 (1993), pp. 313–330.

[64] H. Brendan McMahan and Matthew Streeter. "Adaptive Bound Optimization for Online Convex Optimization". In: *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*. 2010.

[65] George A. Miller. "WordNet: A Lexical Database for English". In: *Communications of the ACM* (1995). URL: http://doi.acm.org/10.1145/219717.219748.

[66] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2016.

[67] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. Chapter 1.4.8. The MIT Press, 2012.

[68] Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. "Path-SGD: Path-Normalized Optimization in Deep Neural Networks." In: *Neural Information Processing Systems (NIPS)*. 2015.

[69] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning". In: *International Conference on Learning Representations (ICLR)*. 2015.

[70] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* (2011).

[71] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. "Improving the Fisher Kernel for Large-scale Image Classification". In: *European Conference on Computer Vision (ECCV)*. 2010. URL: https://www.robots.ox.ac.uk/~vgg/rg/papers/peronnin_etal_ECCV10.pdf.

[72] Jean Ponce et al. "Dataset issues in object recognition". In: *Toward Category-Level Object Recognition*. https://link.springer.com/chapter/10.1007/11957959_2. 2006.

[73] Joaquin Quionero-Candela et al. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[74] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. "Non-convex learning via Stochastic Gradient Langevin Dynamics: a nonasymptotic analysis". arXiv:1702.03849. 2017.

[75] Ali Rahimi and Benjamin Recht. "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning". In: *Advances in Neural Information Processing Systems (NIPS)*. https://papers.nips.cc/paper/3495-weighted-sums-of-random-kitchen-sinks-replacing-minimization-with-randomization-in-learning. 2009.

[76] Esteban Real et al. "Regularized Evolution for Image Classifier Architecture Search". http://arxiv.org/abs/1802.01548. 2018.

[77] Benjamin Recht, Moritz Hardt, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2016.

[78] Benjamin Recht et al. "Do ImageNet Classifiers Generalize to ImageNet?" In: *International Conference on Machine Learning (ICML)*. 2019.

[79] Scott Reed et al. "Generative adversarial text to image synthesis". In: *Proceedings of The International Conference on Machine Learning (ICML)*. 2016.

[80] Aaron Roth and Adam Smith. *Lectures notes "The Algorithmic Foundations of Adaptive Data Analysis"*. https://adaptivedataanalysis.com/. 2017.

[81] Olga Russakovsky et al. "ImageNet large scale visual recognition challenge". In: *International Journal of Computer Vision* (2015). https://arxiv.org/abs/1409.0575.

[82] Shibani Santurkar et al. "How Does Batch Normalization Help Optimization?" In: *Advances in Neural Information Processing Systems 31*. 2018. URL: http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.

[83] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[84] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". https://arxiv.org/abs/1409.1556. 2014.

[85] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013.

[86] Christian Szegedy et al. "Going deeper with convolutions". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1409.4842v1. 2015.

[87] Christian Szegedy et al. "Inception-v4, Inception-Resnet and the impact of residual connections on learning". In: *Conference On Artificial Intelligence (AAAI)*. https://arxiv.org/abs/1602.07261. 2017.

[88] Christian Szegedy et al. "Intriguing properties of neural networks". In: *International Conference on Learning Representations (ICLR)*. http://arxiv.org/abs/1312.6199. 2013.

[89] Christian Szegedy et al. "Rethinking the Inception architecture for computer vision". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1512.00567. 2016.

[90] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[91] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.

[92] Antonio Torralba and Alexei A. Efros. "Unbiased look at dataset bias". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. http://people.csail.mit.edu/torralba/publications/datasets_cvpr11.pdf. 2011.

[93] Antonio Torralba, Rob Fergus, and William. T. Freeman. "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008). https://ieeexplore.ieee.org/document/4531741/.

[94] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* (2004). http://www.cns.nyu.edu/pub/lcv/wang03-preprint.pdf.

[95] Chaowei Xiao et al. "Spatially Transformed Adversarial Examples". In: *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1801.02612. 2018.

[96] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1611.05431. 2017.

[97] Deshraj Yadav et al. "EvalAI: Towards Better Evaluation Systems for AI Agents". 2019. URL: http://arxiv.org/abs/1902.03570.

[98] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. "ShakeDrop regularization". https://arxiv.org/abs/1802.02375. 2018.

[99] Benjamin Z. Yao, Xiong Yang, and Song-Chun Zhu. "Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks". In: *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*. https://link.springer.com/chapter/10.1007/978-3-540-74198-5_14. 2007.

[100] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On early stopping in gradient descent learning". In: *Constructive Approximation* 26.2 (2007), pp. 289–315.

[101] Sergey Zagoruyko. *Torch Blog*. http://torch.ch/blog/2015/07/30/cifar.html. 2015.

[102] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *British Machine Vision Conference (BMVC)*. https://arxiv.org/abs/1605.07146. 2016.

[103] Xingcheng Zhang et al. "Polynet: A pursuit of structural diversity in very deep networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1611.05725. 2017.

[104] Barret Zoph et al. "Learning transferable architectures for scalable image recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. https://arxiv.org/abs/1707.07012. 2018.

[105] Tijana Zrnic and Moritz Hardt. "Natural Analysts in Adaptive Data Analysis". In: *International Conference on Machine Learning (ICML)* (2019).